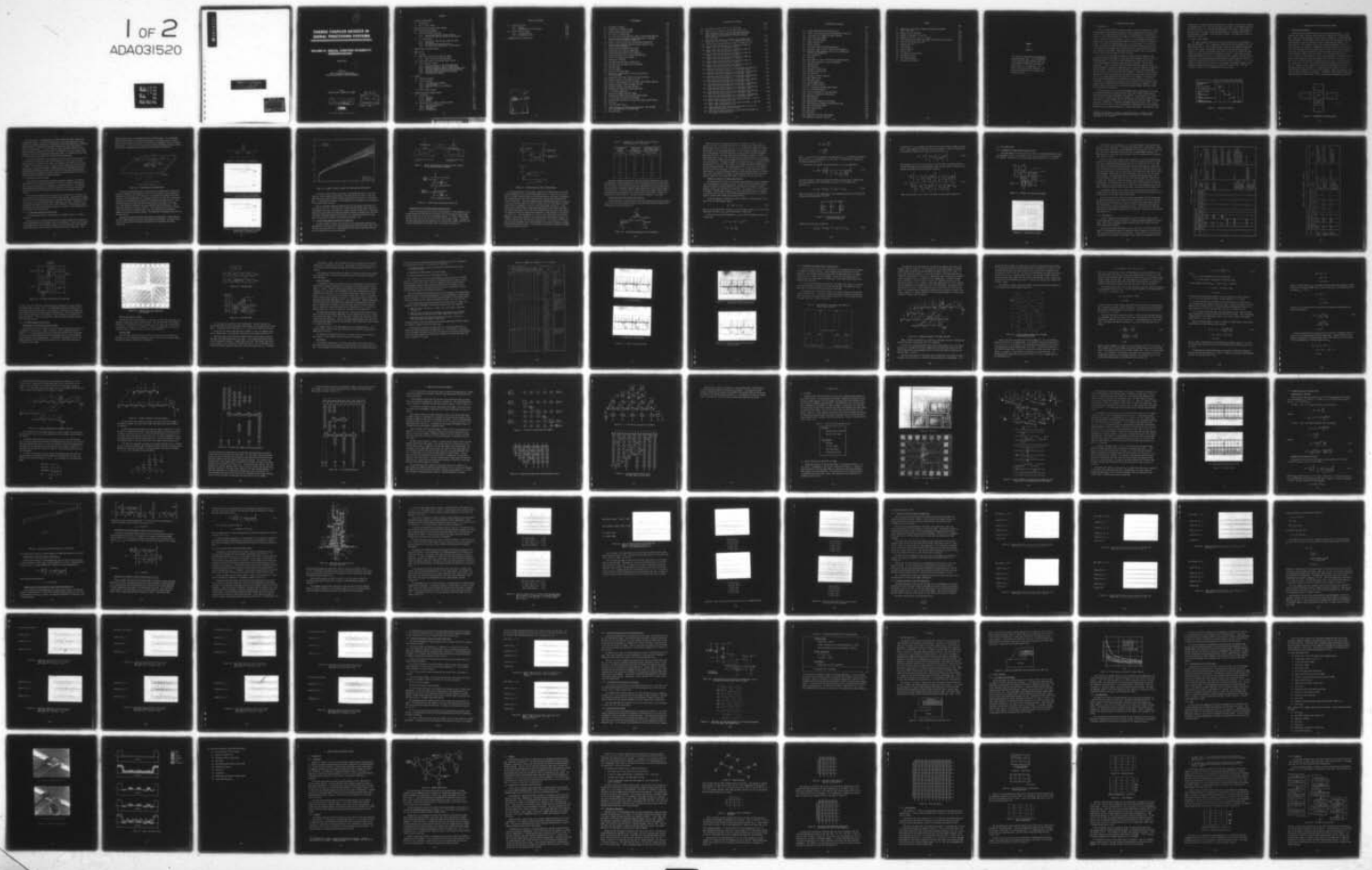
AD-A031 520 TRW SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/5
CHARGE COUPLED DEVICES IN SIGNAL PROCESSING SYSTEMS. VOLUME III--ETC(U)
MAR 76 T A ZIMMERMAN, R J HANDY, R A ALLEN N00014-74-C-0068
UNCLASSIFIED NL

1 OF 2
ADA031520



AD A031520

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC
RECEIVED
NOV 3 1976
A

(4)

CHARGE COUPLED DEVICES IN SIGNAL PROCESSING SYSTEMS

VOLUME III. DIGITAL FUNCTION FEASIBILITY DEMONSTRATION

MARCH 1976

Vol I
782574

Prepared for

NAVAL RESEARCH LABORATORY
NAVAL ELECTRONICS SYSTEMS COMMAND

NAVY
CONTRACT NO.: N00014-74-C-0068

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC
RECEIVED
NOV 3 1976
A

TRW
DEFENSE AND SPACE SYSTEMS GROUP

ONE SPACE PARK • REDONDO BEACH, CALIFORNIA 90278

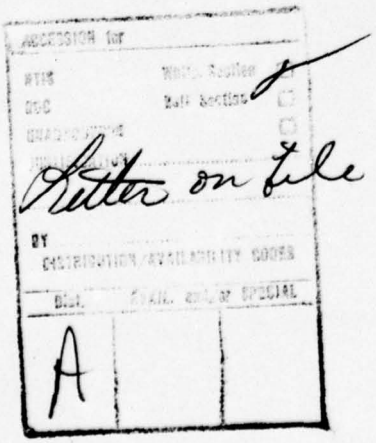
CONTENTS

1.	INTRODUCTION AND SUMMARY	1-1
	1.1 Introduction	1-1
	1.2 Phase 1 Report Summary	1-2
2.	CHARGE DETECTION FOR DIGITAL LOGIC PURPOSES	2-1
	2.1 Operational Requirements	2-1
	2.2 Floating Gate Amplifier Design	2-2
	2.2.1 Floating Gate Amplifier Design on LSM-1	2-5
	2.2.2 Floating Gate Amplifier Design on DP-0 Mask Set	2-5
	2.3 Full Adder Design	2-12
	2.3.1 Original Full Adder Design on Mask Set LSM-1	2-12
	2.3.2 Test Results	2-13
	2.3.3 Full Adder Design on Mask Set DP-0	2-16
	2.3.4 Three-Input Full-Adder Designs on Mask Set DP-1	2-24
3.	COMPUTATIONAL ALGORITHM TRADEOFFS	3-1
4.	MASK SET DP-1	4-1
	4.1 Overview	4-1
	4.2 Detail Description of Two-Input Full Adder	4-1
	4.2.1 Detail Analysis of Two-Input Adder	4-6
	4.3 Detail Description of Three-Input Adder Circuits	4-9
	4.4 Array Functional Testing	4-16
	4.4.1 Functional Testing of the 4-Bit Adder Array	4-16
	4.4.2 Maximum Clock-Rate of the 4-Bit Adder Array	4-16
	4.4.3 Operating Temperature Range of the 4-Bit Adder Array	4-25
	4.4.4 Functional Testing of the 3-Bit Multiplier Array	4-27
	4.4.5 Interfacing the CCD Devices to TTL Drivers	4-27
	4.4.6 Characterization Summary	4-27
5.	PROCESS	5-1
	5.1 Process Description	5-1
	5.2 Special Processes	5-2
	5.2.1 Clean Gate Oxide Technology	5-2
	5.2.2 Ion Implantation	5-3
	5.2.3 Polycrystalline Silicon Technology	5-4
	5.2.4 TEOS	5-4
	5.3 Basic Processing Steps	5-5
6.	SIGNAL PROCESSING INTERFACE STUDIES	6-1
	6.1 Introduction	6-1
	6.1.1 Background	6-1
	6.1.2 Overview	6-1
	6.1.3 Summary	6-3
	6.2 Description of Transparent Interface System	6-3
	6.2.1 Path Matrix Operations	6-4
	6.2.2 Path Selection	6-7
	6.2.3 Flowcharts	6-11
	6.2.4 Path Selection	6-12

PRECEDING PAGE BLANK-NOT FILMED

CONTENTS (Continued)

6.3	Simulation Results	6-13
6.3.1	Example	6-14
6.3.2	Recommendations for Future Work	6-26
6.4	Detailed Design Results	6-27
6.4.1	Crosspoint Switch	6-27
6.4.2	Path Selection Logic	6-35
7.	RECOMMENDATIONS FOR FUTURE WORK	7-1



ILLUSTRATIONS

		Page
1-1	Chronology of Program	1-2
2-1	Accomplishing an AND/OR Function	2-1
2-2	Floating-Gate Test Configuration	2-3
2-3	Basic Test FET Configuration	2-4
2-4	I_{DS} vs V_{DS} with -10 V_{DC} Applied to Gate of Floating Gate Amplifier	2-4
2-5	I_{DS} vs I_{DS} Increase to Where Charge Accumulation is Observed on Gate of FET of Floating Gate Amplifier	2-4
2-6	V_{FGE}NS vs Family of Curves for Floating Gate Configuration	2-5
2-7	Typical Surface Potential Variation Along a Channel for Two Different Gate Arrangements	2-6
2-8	Floating Gate Configuration Used on DPO	2-6
2-9	Floating Gate Test Setup for Ramp Inputs	2-7
2-10	Floating Gate Amplifier with FET Discharge	2-8
2-11	System Equivalent Circuit Floating Gate Amplifier	2-10
2-12	Original Full Adder Block Diagram	2-12
2-13	Three-Input Full Adder	2-12
2-14	Full Adder with Additional Transfer Gate	2-16
2-15	Modified Three-Input Adder with FED Discharge	2-17
2-16	Two-Input Adder	2-18
2-17	Four-Input Adder	2-18
2-18	Output to Three-Input Adder	2-22
2-19	Sum and Carry Output When Two "Ones" Are Applied to Three-Input Adder	2-23
2-20	Output with Three "Ones" Applied to Three-Input Adder	2-23
2-21	Schematic Diagram of Full Adder Test Cell	2-25
2-22	Clock Line Phase Sequence for Full-Adder and Half-Adder Functions	2-26
2-23	Schematic Diagram of the Half-Adder Test Cell	2-30
2-24	Schematic Diagram of Two-Input AND Gate	2-31
2-25	Two-Word 4-Bit Adder Using Three-Input Adders	2-32
2-26	3 X 3 Pipeline Multiplier	2-33
3-1	Four-Bit Full Adder Using Half Adders	3-2
3-2	Two-Word 4-Bit Adder Using Two-Input Adder Circuits	3-2
3-3	4 X 4 Multiplier Using Half (2,2) Adders	3-3
3-4	Two-Word 3-Bit Multiplier Using Two Input Adders and OR Circuits	3-3
4-1	DP-1 Chip	4-2
4-2	Two-Input Adder Circuit	4-2
4-3	Detail Schematic Indicating Surface Potentials, Gate Voltages, and Timing Waveforms of Two-Input Adder	4-3
4-4	Two-Input Adder	4-5

ILLUSTRATIONS (Continued)

		Page
4-5	V_G/ϕ_s Curves for DP1-2 Wafer No. 5 (+10 V Bias)	4-7
4-6	Composite Layout Diagram of the Full-Adder Logic Cell	4-10
4-7	Input and Output Signals of the Two-Input AND Gate Showing that an Output Bit is Generated Only When Both Input Bits are at Logic "1" (Positive Pulses = "1") and a Correct Half Cycle Delay	4-12
4-8	Input and Output Signals of Half-Adder Gate, Showing a Sum Output Each Time A or B Equal "1", but Not When Both Equal "1"	4-13
4-9	Input and Output Waveforms Obtained from Full-Adder Test Cell	4-14
4-10	Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0000 (0)	4-17
4-11	Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0010 (4)	4-17
4-12	Output Signals from the 2-Word, 8-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0100 (8)	4-18
4-13	Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1111 (30) and the b-Word = 1111 (30)	4-18
4-14	Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 0101 (10) and the b-Word = 0101 (10)	4-19
4-15	Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 1000 (16)	4-19
4-16	Three Most Significant Output Bits from the Adder Array Operating with an 11 kHz Clock and Inputs of a = 1100 and b = 0000	4-21
4-17	Three Most Significant Output Bits from the Adder Array Operating with an 11 kHz Clock and Inputs of a = 1100 and b = 0100	4-21
4-18	Three Most Significant Output Bits from the Adder Array Operating with a 50 kHz Clock and Inputs of a = 1100 and b = 0000	4-22
4-19	Three Most Significant Output Bits from the Adder Array Operating with a 50 kHz Clock and Inputs of a = 1100 and b = 0100	4-22
4-20	Three Most Significant Output Bits from the Adder Array Operating with a 100 kHz Clock and Inputs of a = 1100 and b = 0000	4-23
4-21	Three Most Significant Output Bits from the Adder Array Operating with a 100 kHz Clock and Inputs of a = 1100 and b = 0100	4-23
4-22	Three Most Significant Output Bits from the Adder Array Operating with a 175 kHz Clock and Inputs of a = 1100 and b = 0000	4-24
4-23	Three Most Significant Output Bits from the Adder Array Operating with a 175 kHz Clock and Inputs of a = 1100 and b = 0100	4-24
4-24	Output Signals from the 2-Work, 4-Bit Adder Array When a = 0100 (8) and b = 1110 (28) Operated at -65°C	4-26
4-25	Output Signals from the 2-Word, 4-Bit Adder Array When a = 1010 (20) and b = 1010 (20) Operated at -65°C	4-26
4-26	Interconnections and DC Voltage Levels Required to Drive a CCD Signal Processing Device from a TTL Gate	4-28
4-27	Amplitudes and Phase Relationship of Clock and Data Waveforms for TTL to Adder Array Compatibility	4-28

ILLUSTRATIONS (Continued)

		Page
5-1	"Gulch" Formed Under the Polysilicon Film	5-1
5-2	"Gulch" and Steep Polysilicon Step Covered by a TEOS Film	5-2
5-3	Fixed Charge as a Function of Quartz Tube Age	5-3
5-4	Polysilicon Gates (X4000)	5-6
5-5	Steps in the DPl Process	5-7
6-1	Network Organization	6-2
6-2	Network Topology	6-5
6-3	Path Matrix After Interrogating Neighbors	6-5
6-4	Path Matrix After Asking Xl "Who Are Your Neighbors?"	6-6
6-5	Path Matrix After Building a Path to X2 and Asking Him, "Who Are Your Neighbors?"	6-6
6-6	Final Path Matrix	6-7
6-7	Stack Word	6-8
6-8	Stack After X4's Row in Path Matrix Has Been Connected	6-8
6-9	Stack After Scanning Xl's Row in Path Matrix	6-8
6-10	Completed Stack	6-9
6-11	Path Threading	6-9
6-12	Threading Alternate Stack	6-10
6-13	Path Matrix Construction	6-11
6-14	Path Selection	6-12
6-15	Flowchart of Overall Operation	6-13
6-16	Basic SONAR System	6-14
6-17	Simulated System	6-15
6-18	Improved SONAR Network	6-25
6-19	Basic Crosspoint Switch	6-27
6-20	8 x 4 Crosspoint Detailed Logic Diagram	6-29
6-21	16 x 8 Crosspoint Switch	6-33
6-22	Latch for Expansion of Trunk Busy Signal	6-35
6-23	Circuit for Expansion of Busy Signal	6-35
6-24	Node Controller	6-36
6-25	Line Interface	6-37
6-26	Detailed Path Selection Logic Diagram	6-39
6-27	Simplified Block Diagram of Path Selection Logic	6-41
6-28	A Row from the Path Matrix	6-42
6-29	Node i Connectivity	6-43
6-30	Stack Word	6-44
6-31	Path Stack	6-44
6-32	Sequential Controller State Diagram	6-46
6-33	Sequential Controller Flowchart	6-48

TABLES

		Page
2-1	Comparison of Calculated vs Measured Floating Gate Voltages for R = 10^{13} ohms	2-8
2-2	Summary of Lots 1 Through 6	2-14
2-3	Summary of Lots NAV-7, -10 and -11	2-15
2-4	Summary of Lots DPO-1, -2, -3, -4 and -5	2-21
2-5	Truth Tables for Full-Adder, Half-Adder and AND Gate Logic Functions	2-24
4-1	Circuits and Test Devices on DP-1	4-1
6-1	Proposed Path File	6-17
6-2	Trace by Node	6-19
6-3	Trace by Message	6-20
6-4	Trace by Time	6-21
6-5	Throughput Time Distribution	6-22
6-6	Path Establish Time	6-22
6-7	Path Length Frequency	6-23
6-8	Transmit Type Frequency	6-23

FOREWORD

TO

VOLUME III

This report has been prepared by TRW Defense and Space Systems Group. The work summarized in this report has been performed under Contract N00014-74-C-0068 and was directed for the Navy by Dr. D. F. Barbe and Dr. W. D. Baker of the Naval Research Laboratory. The period of performance was from January 1975 to February 1976.

1. INTRODUCTION AND SUMMARY

1.1 INTRODUCTION

In 1973, the Naval Research Laboratory issued a request for quotation for a study program aimed at defining and analyzing those areas of application of charge coupled devices (CCDs) in signal processing systems. The broad objective of the RFQ was to initiate a study that would examine the impact of CCD technology on signal processing systems. Implicit in such a statement, of course, is the requirement to determine those areas of signal processing systems where the use of CCDs offers an economic advantage. The extent of that advantage, that is to say the impact, can then be projected. Naturally, the projection cannot be made in terms of dollars and cents, but is best made by direct comparison of identical functions realized with CCDs and any other appropriate technology. Under these conditions, numbers such as speed, power, and parts count can be tabulated and cross-correlated.

As a result of the proposal submitted to the Naval Research Laboratory, TRW embarked on a study of the impact on signal processing systems of the use of CCDs in the digital domain. The results of that study have been issued under the title "Charge Coupled Devices in Signal Processing Systems; Volume I: Digital Signal Processing."* Briefly stated, the study indicated that digital CCDs combine the inherent advantages of any digital technology (such as high noise immunity, freedom from device/parameter variations, and stable operating conditions) with the advantages peculiar to CCDs (such as high density and low power). In addition, digital CCDs are best suited to those signal processing applications where the signal flow can be carried out in a pipelining fashion requiring little or no feedback; this permits relatively high data throughput to be accomplished with the relatively low CCD clock frequencies. Not surprisingly, the impact is most dramatic in those situations where a large number of functions and/or high computational accuracy is demanded. A large number of such instances occur in existing and projected systems; these were identified and analyzed in some detail.

At the conclusion of the study, TRW recommended that an experimental verification be carried out that would go beyond the basic device work already accomplished and would demonstrate the real advantages of the approach. The realization of a digital CCD fast Fourier transform on a chip was selected as a useful vehicle; additionally this function, properly implemented, is quite flexible and suited to a number of diverse situations. Accordingly, a technology development program was entered into. The objective of the first phase is the investigation and characterization of the fundamental building blocks that would be employed in a typical application. The objective of the

*Available from the National Technical Information Services; a companion report "Charge Coupled Devices in Signal Processing Systems; Volume II: Analog Signal Processing" is also available.

second phase is to develop those building blocks into larger circuit functions suitable for implementing the FFT. And the third phase is directed at an FFT demonstration unit specifically aimed at a to-be-determined application. This report is being issued at the end of the first phase. The second phase duration is 13 months; the duration of the third phase is dependent on the application selected. The chronology of events is summarized in Figure 1-1.

1.2 PHASE 1 REPORT SUMMARY

This report contains an overview of the entire program and a brief statement of goals and approaches. This is followed by a discussion of the development of the full adder circuit function. The original concept is explained and subsequent alterations to the original layout are described; both two and three input adders are treated (Section 2). There are some hardware implications in the several computational algorithms that can be used and these are examined in Section 3. The primary test mask that was designed during Phase 1 is presented along with a summary of the test results in Section 4. The process sequences being employed to produce these devices are explained, and cross-sectional views of the devices are given in Section 5. This is followed by a presentation of the results of a study made to determine a method of interconnecting a number of the projected FFT chips into a single system. The report concludes in Section 7 with a recommendation for future work.

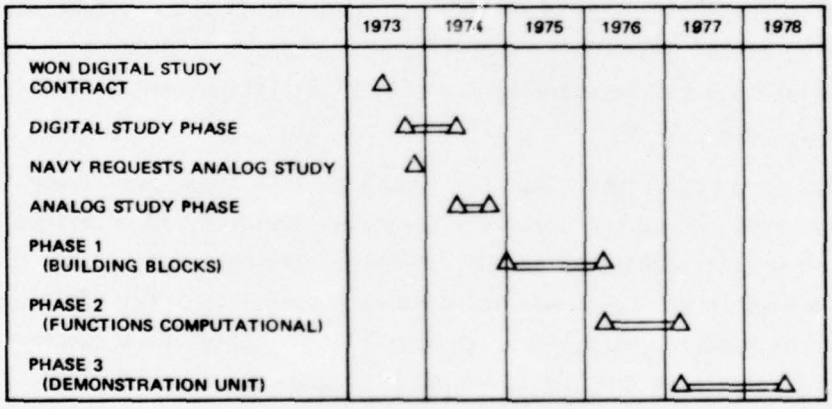


Figure 1-1. Chronology of Program

2. CHARGE DETECTION FOR DIGITAL LOGIC PURPOSES

2.1 OPERATIONAL REQUIREMENTS

Several distinctly different ways exist to perform logic with CCD's. In one approach, two or more individual charge packets are combined into a single storage area. Then, by making measurements on this result, information is derived about the original packets. This technique is useful only insofar as the information derivable from the addition cannot be obtained from the isolated individual packets. As a simple example of this approach, consider the electrode arrangement of Figure 2-1. This figure shows the conceptual arrangement for performing an AND and an OR function. If all the electrodes can store an equal amount of charge, then when the contents of A and B are simultaneously transferred to the common gate covering the confluence of the two channels, several results are possible. If A and B are both empty (i.e., A = 0 = B) then, of course, there will be no charge transferred to the common gate and with the next transfer the electrode labeled C will also contain no charge (C = 0). Now if A or B are storing charge, C eventually will also do so. If A and B both contain charge, clearly C will too, after the contents of the common gate are emptied into it. However, since the common gate can hold only the charge of either A or B, the excess charge must flow under D. By examining the truth tables of C and D, it is clear that both an AND and an OR function are generated. Other more complicated functions can also be realized in this manner. The identity of the original digital data which provides the A and B values is lost in the process of performing the function.

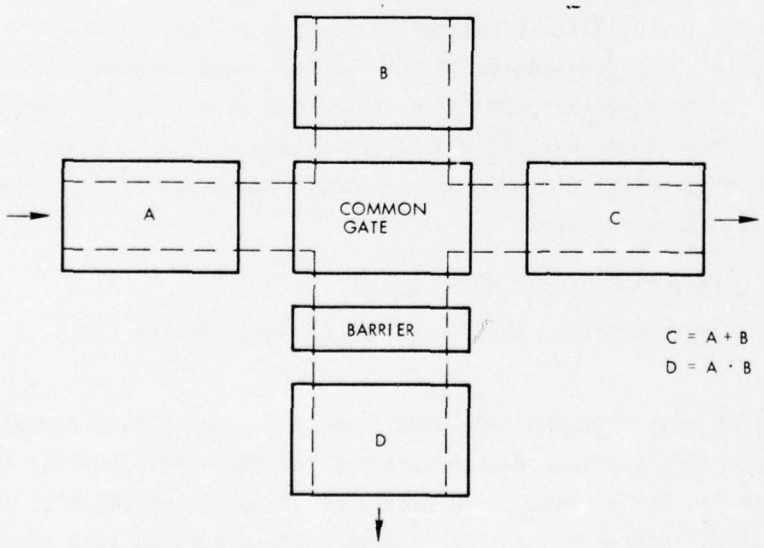


Figure 2-1. Accomplishing an AND/OR Function

This loss of data is not necessary, and other schemes exist which preserve the original data streams. In order to perform logic (or any other operation) on data streams and leave them undisturbed, a nondestructive sensing technique must be employed. Then the presence or absence of charge may be detected and some action taken as a result of this detection. The action taken may involve switching a MOS FET from one state to another or simply controlling flow in another CCD as a result of this charge detection in the CCD containing the original data stream. The floating diffusion and/or floating gate are two common means of performing such detection.

The circuits described in this report make use of both of the above types of circuit functions to achieve the overall desired operation; in some cases individual charges packets are combined, and in other cases nondestructive sensing is all that is needed. Which technique is used is simply a matter of the best way to perform the task at hand. The basic philosophy is to begin with the CCD characteristics, their advantages and limitations, and design to achieve the "black box" function desired. Merely mimicking the approach of a similar function constructed using other than CCD technology generally produces a cumbersome solution.

2.2 FLOATING GATE AMPLIFIER DESIGN

A key element in the design of CCD digital processing systems is the floating gate amplifier. The floating gate amplifier is involved in the logic decision process, and its proper operation is essential to the overall function. The floating gate nondestructively detects the presence or absence of charge in one location and transmits and utilizes the information to control the flow of charges at another location.

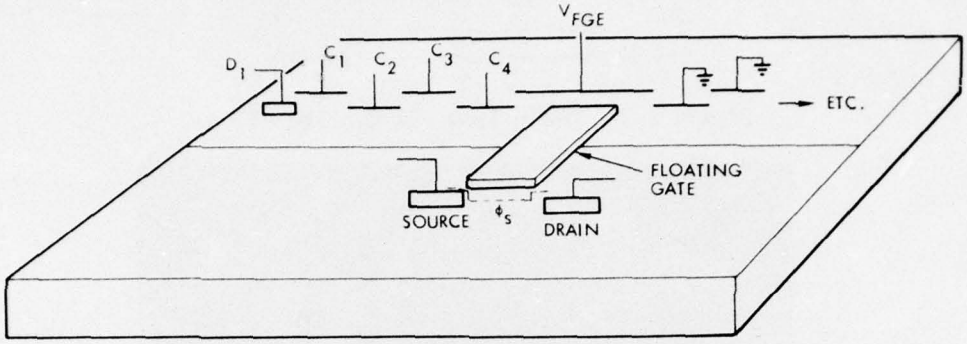
Thus the floating gate must be sensitive to signal charge and its variations but at the same time be insensitive to all parasitic type charges. These parasitic charges may be intrinsic to the processing cycle, or induced image charges generated by charges which may exist in the atmosphere or may be deposited on surfaces. Theoretical and experimental considerations indicate that a FET attached to a floating gate structure would permit the proper floating gate performance. Evolution of the floating gate amplifier design is discussed below.

2.2.1 Floating Gate Amplifier Design on LSM-1

The floating gate amplifier configuration, as developed on LSM-1, is shown in Figure 2-2.

During test of this floating gate amplifier, the gate voltage versus surface potential measurements on the same device were not reproducible. Further test and analysis revealed that for drain voltages in excess of 10 volts (relative to the floating gate voltage) charge injection onto the floating gate structure took place. This

charge injection takes place between the drain and floating gate. It is postulated that hot electrons which are available at the drain are accelerated to the floating gate structure because of the electric field that exists between the drain and floating gate structure. Since the FET drain and source diffusion breakdown voltage was greater than 20 volts, breakdown was not involved in the threshold shift mechanism.



V_{C_1} = -39v V_{C_3} = -37v V_{D_1} = -18v (VARIABLE)

V_{C_2} = -30v V_{C_4} = -28v

V_{FGE} = 0 to-35v (VARIABLE)

INJECT 1μA INTO SOURCE WHILE V_D = -10v

Figure 2-2. Floating-Gate Test Configuration

To verify that charging was due to injection on the floating gate structure and not due to trapping at the interfaces, a test FET on the same chip was configured to represent the floating gate FET (Figure 2-3). By applying the voltages to the gate and then lifting the gate probe off, a charge was left on the gate; this charge should exhibit the same characteristics as those of the floating gate amplifier FET. Tests indicated that when the drain voltage was more negative with respect to the gate ($\Delta V \approx 10V$) charge was injected on the gate, making the gate more negative. These results are depicted in Figures 2-4 and 2-5. Since the gate of the test FET is accessible, while that of the floating gate is not, the stored charge was completely removed by connecting the gate to ground. This experiment was repeated many times with the same results.

Applying this theory to floating gate amplifier configuration, charge deposited onto the floating gate structure was controllable and predictable. The gate voltage versus surface potential curves were reproducible and in agreement with theory (Figure 2-6).

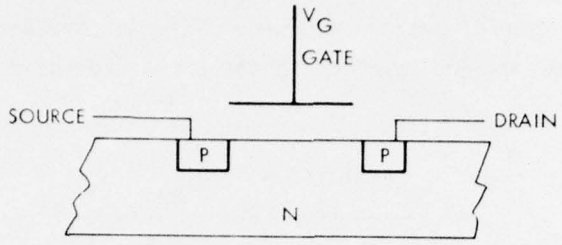


Figure 2-3. Basic Test FET Configuration

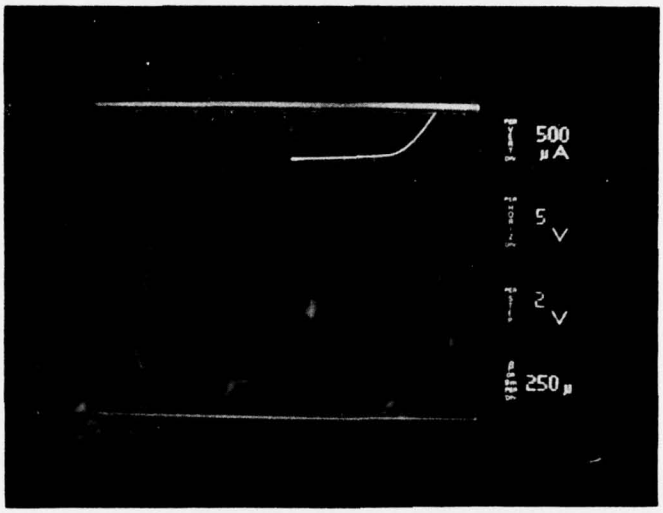


Figure 2-4. I_{DS} vs V_{DS} with -10 V_{DC} Applied to Gate of Floating Gate Amplifier

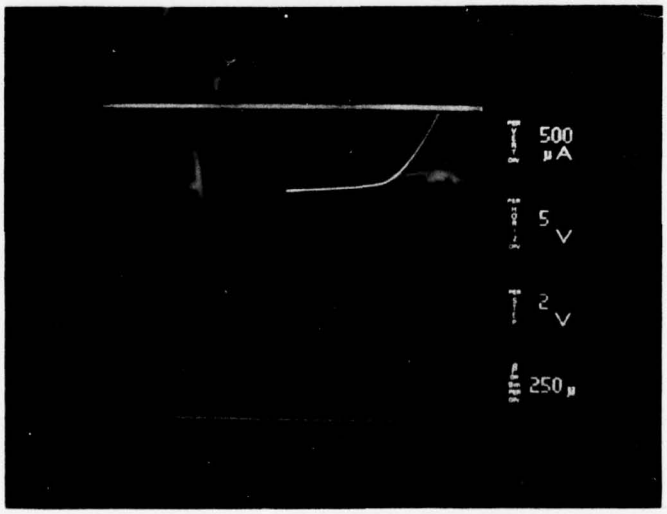


Figure 2-5. I_{DS} vs I_{DS} Increase to Where "Charge Accumulation is Observed on Gate of FET of Floating Gate Amplifier

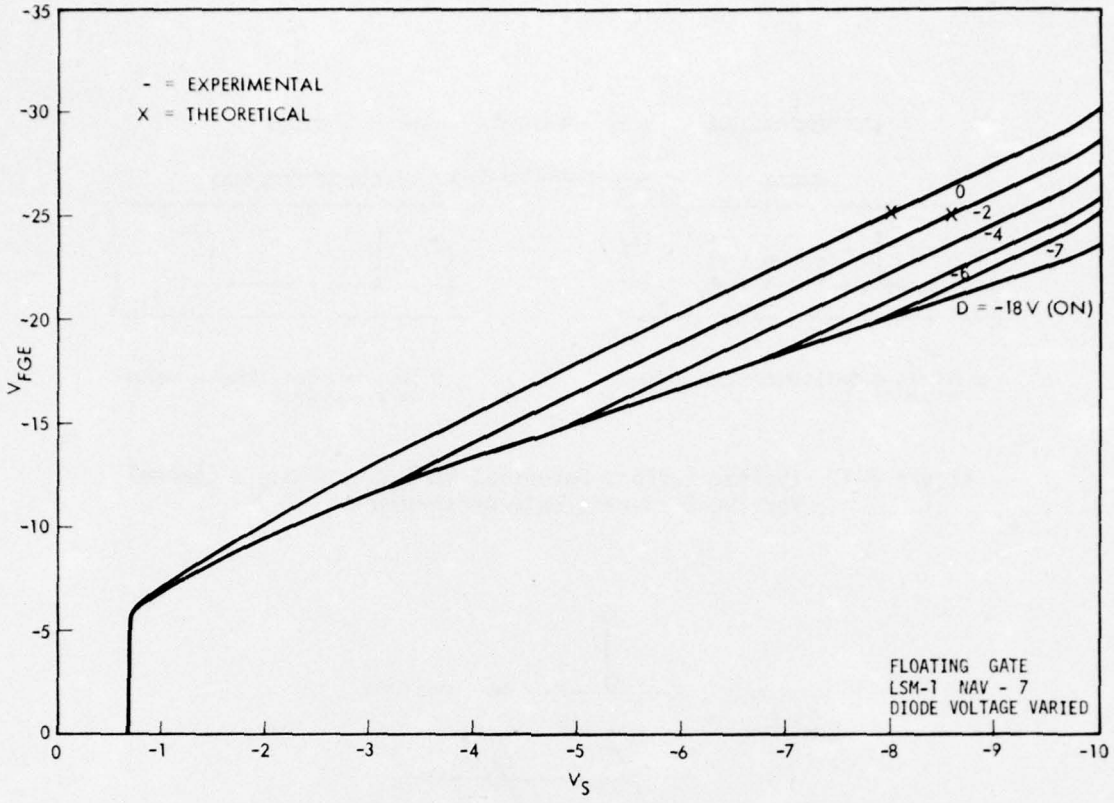


Figure 2-6. V_{FGE}NS vs Family of Curves for Floating Gate Configuration

In addition to this problem, during test of the three-input adders on the LSM-1 mask set, charge voltages were observed on the floating gate structure. Tests indicated that during wafer processing, charge accumulation was occurring to the floating gate. To eliminate both problems a FET discharge circuit was added to allow discharge of the floating gate structure.

Finally, one other problem was observed with the floating gate configuration as indicated in Figure 2-2. The bias gate is larger than the floating gate; this generates potential variations under the gates along the channel (Figure 2-7a). These bumps produce two regions which are capable of trapping carriers and holding them, and therefore impair the proper operation of the device. To correct this situation the bias gate was made smaller related to the floating gate (Figure 2-7b).

2.2.2 Floating Gate Amplifier Design on DP-0 Mask Set

The above concept of adding a FET discharge to the floating gate structure was incorporated in the design of the floating gate amplifier and three-input adder on the DP-0 mask set (see Figure 2-8). Tests performed on the circuits indicated that they were not susceptible to charge accumulation.

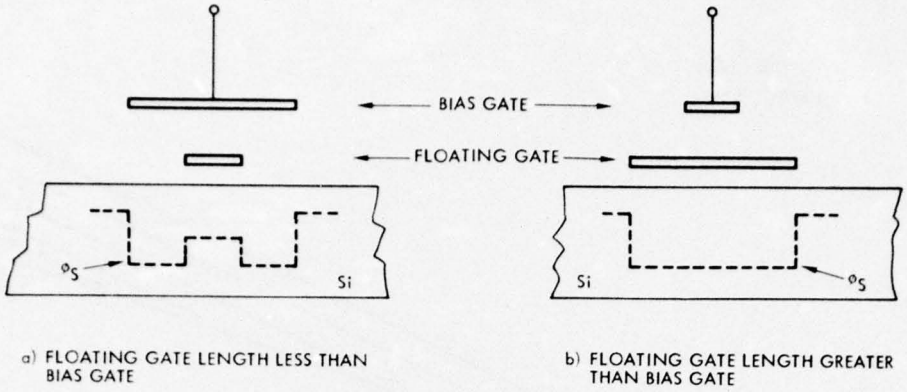


Figure 2-7. Typical Surface Potential Variation Along a Channel for Two Different Gate Arrangements

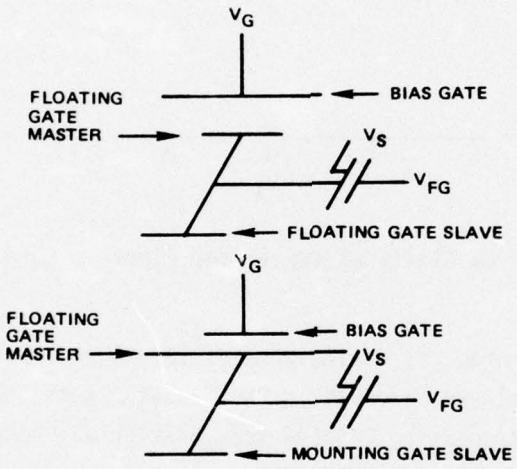


Figure 2-8. Floating Gate Configuration Used on DPO

Determination of FET Charge/Discharge Characteristics

Since the floating gate with reset FET now possesses a finite impedance to substrate (through the reset FET drain diffusion), the control-gate-to-floating-gate transfer function is of the form of a highpass network. As a result, ramp voltages applied to the control gate will result in dc floating gate voltages. A diagram of the test setup along with the equation for the floating gate voltage, resulting from a ramp voltage applied to the control gate, are provided in Figure 2-9.

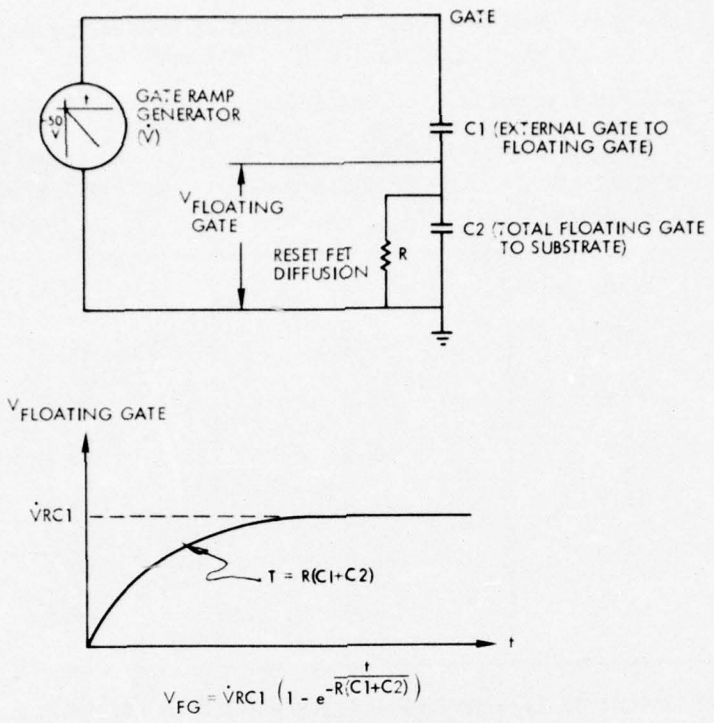


Figure 2-9. Floating Gate Test Setup for Ramp Inputs

The surface potential of the channel region of the floating gate FET was determined by measuring the FET source voltage required for a 1 μA drain to source current. To establish baseline data on the floating gate FET, the reset FET attached to the floating gate was driven into conduction and a voltage ramp applied directly to the floating gate. The channel region surface potentials were measured (using the μA criteria described above) for several different values of floating gate voltages. These results are presented in the first and second columns of Table 2-1. After obtaining this baseline data, voltage ramps were applied to the control gate (with the floating gate reset FET gate and drain grounded) and V/sec rate ($\dot{V}$) and duration (t) adjusted to obtain the same surface potentials as those obtained in the baseline run (Table 2-1). From knowledge of $\dot{V}$, t, and C1 and C2 (calculated from the device geometry), the remaining variable in the floating gate voltage equation (Figure 2-9) is the resistance of the reset FET drain diffusion (R). If a value of 10^{13} ohms was used for R, good agreement was obtained between the calculated floating gate voltage and the measured baseline values. The calculated floating gate voltages are provided in the third column of Table 2-1.

Table 2-1. Comparison of Calculated vs Measured Floating Gate Voltages for R = 10^{13} ohms

Floating Gate Surface Potential (ϕ_S) (Volts)	Floating Gate Voltage by Direct Measurement (V_{FGD}) (Volts)	Calculated Floating Gate Voltage from Ramp Input R=10^{13} Ω (V_{FGC}) (Volts)
-0.6	-1.8	-2
-1.0	-2.3	-2.2
-1.5	-2.8	-2.7
-2.0	-3.5	-3.2
-3.0	-4.8	-4.4
-4.0	-6.0	-6.5
-5.0	-7.0	-7.7

Subsequent discharge time-constant tests and sinusoidal phase shift tests were performed which provided agreement with the 10^{13} ohm value for reset FET drain diffusion resistance. These test results have thus determined the drain diffusion resistance and yielded good agreement between device measurements and calculations based on the present model. Results to date for pulsed control gate voltages indicate that a pulse amplitude of approximately twice that predicted by the model is required to threshold the floating gate FET. The cause of this discrepancy is not known.

Floating Gate Amplifier Design on DP-1 Mask Set

Further theoretical analysis revealed that the performance and physical configuration of the floating gate amplifier would be achieved by designing and operating the floating gate structure (Figure 2-10).

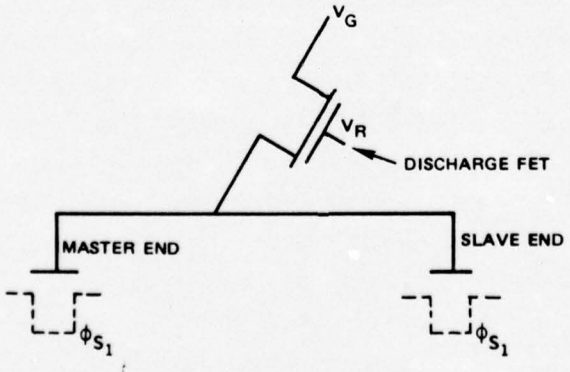


Figure 2-10. Floating Gate Amplifier with FET Discharge

Operation of this floating gate structure is as follows. At time t_o, V_R is applied to the FET; its polarity turns the FET on, applying the voltage V_G to the floating gate structure. At time t_1, ($t_1 = t_o + \Delta t$), the FET is turned off by removing voltage V_R. The floating gate structure is now floating and charged to a potential V_G and produces the surface potentials ϕ_{s1} and ϕ_{s2} under the gates designated master and slave end. The master end detects the presence or absence of charge Q being dumped into its well. The slave end varies its surface potential ϕ_{s2} as a function of the amount of charge Q being dumped into the master well. This is accomplished in the following way. As charge is dumped in the master side well, it charges the surface potential ϕ_{s1} to a new level, which in turn changes the charge distribution on the floating gate structure. This charge distribution in turn varies the surface potential ϕ_{s2} under the load gate. This variance in ϕ_{s2} is then used as part of the logic function. Due to the inherent inversion in this operation, charge is allowed to flow in the slave channel if no charge is detected under the master gate, and slave channel charge flow is prevented when charge is detected under the master gate.

Theoretical Analysis of Floating Gate Structure

The magnitude of the variance in slave gate surface potential plays an important role in the operation of the logic circuits. A close form expression which directly relates the changes of surface potential at the Master End to changes of surface potential at the slave load end is very desirable. The expression will be a function of changes in floating gate voltages (ΔV_G), any initial charge originally contained in the master well (fat zero), initial gate voltage applied at time τ_o, V_G, and the amount of charge Q being dumped into the detector well.

The design equation is obtained by writing Kirckhoff's voltage equation for the gate, oxide, and semiconductor system:

$$V_G - V_{FB} = \phi_s + V_{ox} \tag{2.1}$$

where V_G is the applied gate voltage, V_{FB} is the flat band voltage, V_{ox} is the voltage drop across the oxide, and ϕ_s the surface potential under gate.

Solving for the explicit dependence of ϕ_s on the device parameters and bias conditions gives,

$$\phi_s = V + V_o - \left(V_o^2 + 2\,VV_o\right)^{1/2} \tag{2.2}$$

$$V = V_G - V_{FB} - \frac{q_o}{C_{ox}}$$

$$V_{FB} = \phi_{MS} - \frac{Q_{ss}}{C_{ox}}$$

$$V_o = \frac{e\epsilon_s N_A}{C_{ox}^2}$$

where ϵ_s is the dielectric constant of the semiconductor, C_{ox} is the oxide capacitance per unit area, ϵ_x is the permittivity of the oxide, e is the electronic charge, and the doping concentration is N_A.

The change in the detector end surface potential ϕ_{s_1} can be expressed as

$$\Delta\phi_{s_1} = \left(\Delta V_G - \frac{\Delta Q}{C_{ox_2}}\right)\left\{1 - \frac{1}{\left[1 + \frac{2(V_G - V_{FB}) - \frac{q_o}{C_{ox2}}}{V_o}\right]^{1/2}}\right\} \qquad (2.3)$$

The other functional relationship between the master and slave gates is derived from their interconnection which is depicted in Figure 2-11 and is expressed by equation (2.4)

$$\phi_{s_2} C_{ox_2} = V_G \left(C_{ox_2} + C_T + C_{ox_4}\right) - \phi_{s_1} C_{ox_4} \qquad (2.4)$$

where C_T includes all other capacitances of the floating gate structure, including the source diffusion capacitance of the FET.

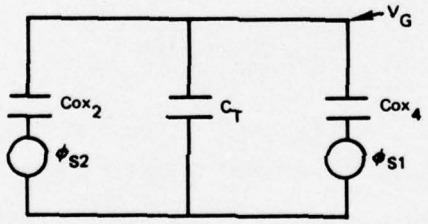


Figure 2-11. System Equivalent Circuit Floating Gate Amplifier

Equation (2.4) can be put in the form

$$\Delta\phi_{s_2} C_{ox_2} = \Delta V_G \left(C_{ox_2} + C_T + C_{ox_4}\right) - \Delta\phi_{s_1} C_{ox_4} \qquad (2.5)$$

In equation (2.5) $\Delta\phi_{s_2}$ represents the change in the surface potential under slave gate which is always empty of charge during its normal operation. Thus the slave gate can be represented by equation (2.3) with q_o set to zero and $\Delta Q = 0$:

$$\Delta\phi_{s_2} = \Delta V_G \left\{ 1 - \frac{1}{\left[1 + \frac{2\,(V_G - V_{FB})}{V_o} \right]^{1/2}} \right\} \tag{2.6}$$

Using equations (2.6), (2.5), (2.3) and $q_o = 0$ (no fat zero), the desired result of finding the changes of the surface potential under the slave end as a function of the charges being dumped into the master end is given as

$$\Delta\phi_{s_2} \frac{\frac{\Delta Q}{C_{ox_2}} \left\{ 1 - \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_2}\right)}{V_o} \right]^{1/2}} \right\} \left\{ 1 - \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_4}\right)}{V_o} \right]^{1/2}} \right\}}{- \frac{C_{ox_4}}{C_{ox_2}} \left\{ \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_4}\right)}{V_o} \right]^{1/2}} \right\} - \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_2}\right)}{V_o} \right]^{1/2}} - \frac{C_T}{C_{ox_2}}} \tag{2.7}$$

Where the subscripts 2 and 4 refer to the master and slave gates respectively.

2.3 FULL ADDER DESIGN

2.3.1 Original Full Adder Design on Mask Set LSM-1

The block diagram of the three-input full adder as originally designed on a mask set designated as LSM-1 is indicated in Figure 2-12, and Figure 2-13 is a photograph of a device. The operation of the three-input adder (Figure 2-12) is as follows.

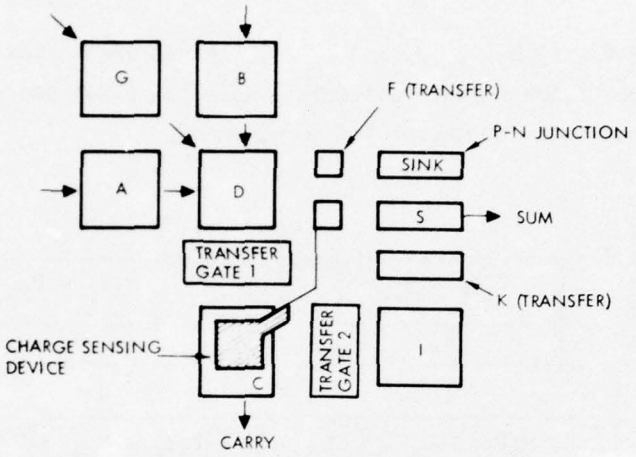


Figure 2-12. Original Full Adder Block Diagram

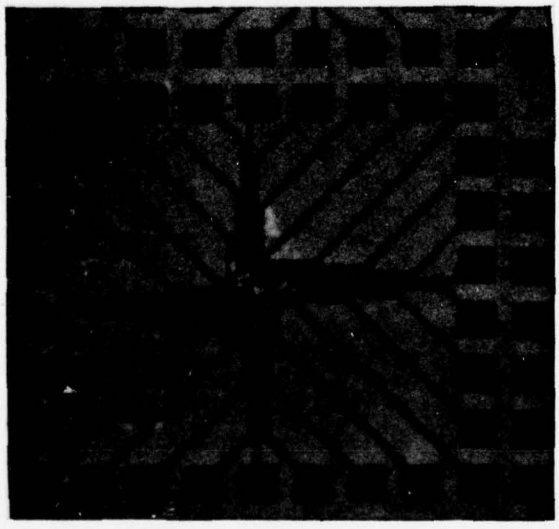


Figure 2-13. Three-Input Full Adder

Three digital inputs designated A, B, and G generate equal charge packets under gates A, B, and G respectively. The charge under these gates is simultaneously dumped into the potential well under the gate designated first storage gate. The first storage gate, carry gate, intermediate storage gate, and sum gate generate potential wells which are equal to each other, and also equal to each of the potential wells generated under gates A, B, and G.

During the time interval when the shift registers A, B, and G are dumping charge into the first storage gate, transfer gates 1 and 2 are biased such that if all gates A, B, and G originally contained charge, then each of the gates (first storage gate, carry gate, and intermediate storage gate) will have an equal amount of charge under them. The operation of the three-input adder is based on the fact that the carry gate will have charge under it only if at least two of the three input gates contain charge. The intermediate storage gate will have charge under it only if all the input gates contain charge. The charge-sensing device, called the floating gate master end, detects the presence of charge under the carry gate and translates this information to its slave end. By means of capacitive coupling, the carry gate is used to bias the floating gate. The floating gate at the slave end will allow charge to flow from the first storage gate to the sum gate only if there is no charge detected under the carry gate; if charge is detected under the carry gate, the slave end prevents charge flow.

An undesirable feature of this design is that the sum gate must be clocked to prevent charge flow from the first storage gate until enough time has elapsed to allow any overflow to fill the well under the carry gate.

This can be corrected by inserting a control gate between the first storage gate and the floating gate slave end, which will not allow charge to flow to the sum gate during the time interval gates A, B, and G are dumping their charge packets into the first storage gate. This will allow time for charge, if available, to be detected under the carry gate by the floating gate master end. This new design of the three-input adder was developed and tested on DP-0, is depicted in Figure 2-14, and discussed in paragraph 2.3.2.

2.3.2 Test Results

Table 2-2 summarizes the characteristics of lots 1 through 6; these contain floating gate amplifiers and three-input full adders which were processed on the LSM-1 mask. These circuits exhibited unstable threshold shifts and indicated excessive boron and phosphorous implant.

Testing of the three-input adders of lots 7, 10, and 11 of the LSM-1 mask (see Table 2-3) indicated the presence of charge sufficient to produce large positive voltages in the poly floating gate (up to 80 volts). In most cases this charge could be eliminated only by applying a large negative pulse voltage of approximately 90 volts

Table 2-2. Summary of Lots 1 Through 6

Lot No.	Major Processing Characteristics						Device/Circuit Characteristics						
	Poly/Oxide Thickness		Metal/Oxide Thickness		Field Oxide		Test FETs (Threshold Voltages)				Floating Gate Amplifiers	Full Adders	Remarks
	Wet	Dry	Wet	Dry	Wet	Dry	Poly/ Gate	Metal/ Gate	Poly/ Field	Metal/ Field			
1		1000Å		TEOS 2000Å	X		-1.5	-4.4	--	--	Tested and found to be functional	Tests attempted, floating gate region in depletion mode	Depletion mode believed to be due to later discovery of charge injection onto the floating gate
2		1000Å		TEOS 2000Å			-1.7	-4.8	--	--	Tested and found to be functional	Tested and found to be partially operable	Examination of this full adder result lead to the discovery of a basic design geometry problem in the full adder
3		500Å		TEOS 2000Å			--	--	--	--	--	--	Lot not functional, suspect excessive boron and phosphorous implant energy
4		500Å	2000Å				--	--	--	--	--	--	Lot not functional, suspect excessive boron and phosphorous implant energy
5		500Å	2000Å				-5	-11	--	--	Tested and some functional units found. Most units exhibit unstable threshold characteristics	Not tested	Threshold instability later found to be due to floating gate charge injection. Results from functional units used to refine floating gate computer model.
6		500Å	2000Å				-3.2	-8	--	--	Tested and some functional units exhibit unstable threshold characteristics	Tested and found to have unstable threshold characteristics	Same as Above

Table 2-3. Summary of Lots NAV-7, -10 and -11

Lot No.	Major Processing Characteristics						Device/Circuit Characteristics						
	Poly/Oxide Thickness		Metal/Oxide Thickness		Field Oxide		Test FETs (Threshold Voltages)				Floating Gate Amplifiers	Full Adders	Remarks
	Wet	Dry	Wet	Dry	Wet	Dry	Poly/ Gate	Metal/ Gate	Poly/ Field	Metal/ Field			
NAV-7 Wafer 1 2	1000Å		2000Å		x		-3.0 V	-7.5 V	--	--	Threshold (V_T) = -6.0 V_S and V_G curve ok	Bad	All full adders bad because floating gate charge too high ≈+80 V
NAV-11 Wafer 26 27 28	1000Å		2000Å		x		-4.0 V	-6.5 V	--	--	Threshold (V_T) = -8.0 and -10 V V_S and V_G curve ok	Bad	All full adders bad floating gate charge too high ≈+80 V
NAV-11 Wafer 26 27 28	1000Å		2000Å		x		-3.0 V	-5.0 V	--	--	V_S and V_G curve ok. After mounting dice on cap V_S and V_G curve can't repeat	None operating	All full adder floating gate charge too high, negative voltage

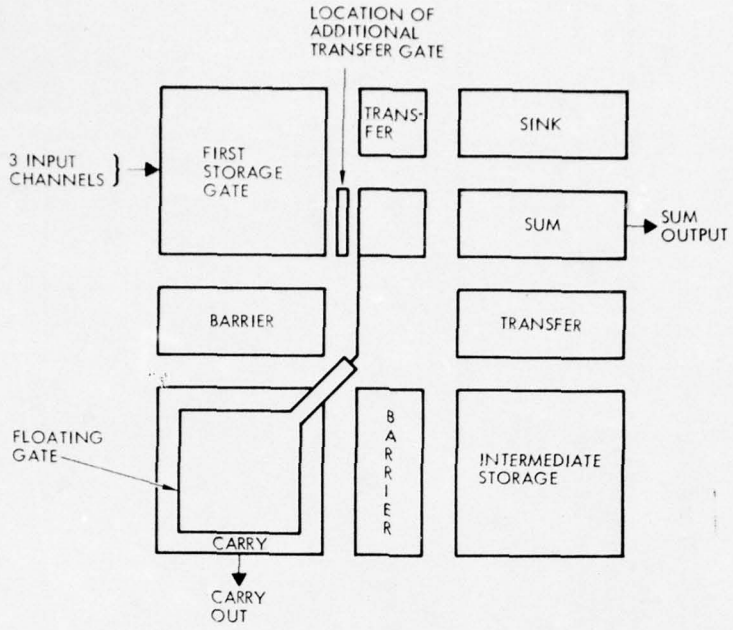


Figure 2-14. Full Adder with Additional Transfer Gate

in value. Once this charge was eliminated the gate voltage versus surface potential measurements were not stable. That is, the measurements shifted in different directions when taken on the same device. This instability indicated that the large voltage discharge through the oxide was breaking down the oxide. Since the floating gate had no direct current path to the substrate, it was not possible to discharge any charge accumulation. This was accounted for in the DP-0 design by adding a discharge FET to the floating gate.

2.3.3 Full Adder Design on Mask Set DP-0

Modification to Existing Three-Input Full Adder

The modified original full adder circuit block diagram, depicting the location of the additional transfer gate, is given in Figure 2-14 and a photograph appears in Figure 2-15. This gate decouples the surface potential under the first storage gate from that of the floating gate and allows an independent choice of the dc potential on the bias gate over the floating gate. This gate and the addition of the FET discharge capability to the floating gate structure led to a more stable operation of the three-input full adder.

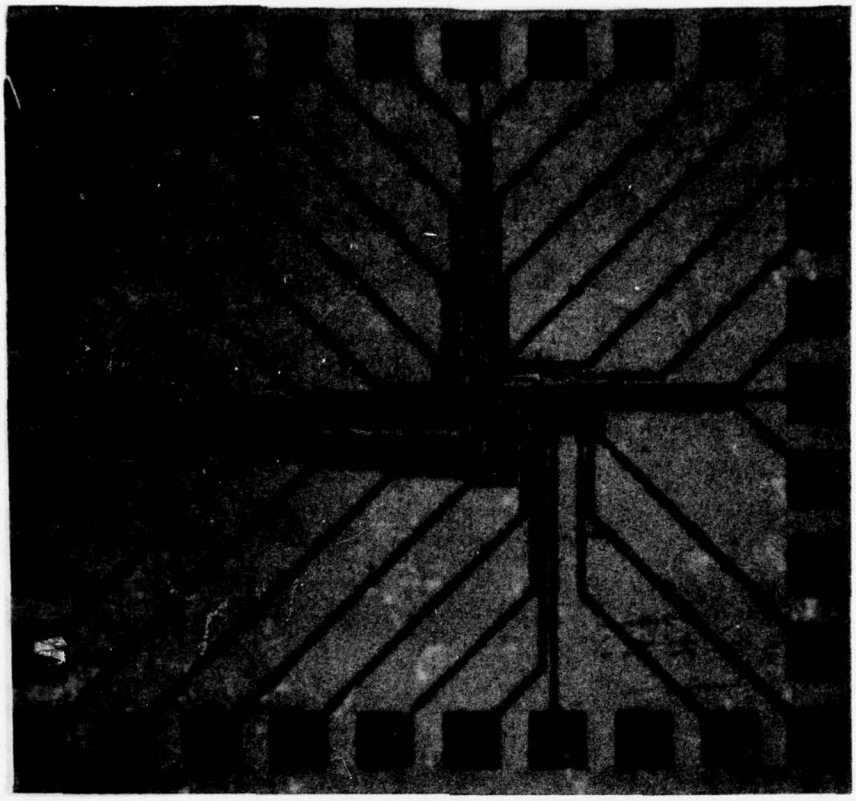


Figure 2-15. Modified Three-Input Adder with FED Discharge

Two and Four-Input Full Adders

During evaluation and implementation of the basic CCD three-input full adder, two additional unique adder design concepts evolved: the two-input adder, which behaves similarly to a conventional digital half adder, and a four-input adder which features an automatic "look ahead carry" capability. The basic configuration of the two and four-input adder are shown in Figures 2-16 and 2-17 respectively.

Two-Input Adder

This full adder differs from the original basic three-input full adder by the elimination of the intermediate storage gate, transfer gate, and control gate between the carry out gate and intermediate storage gate.

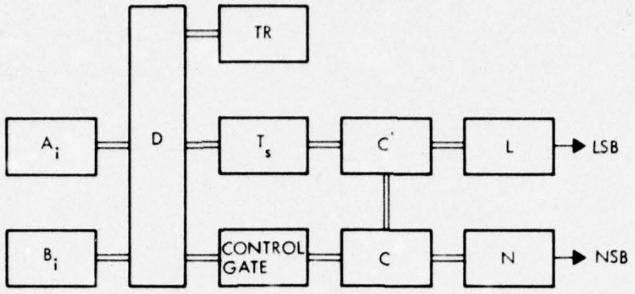


Figure 2-16. Two-Input Adder

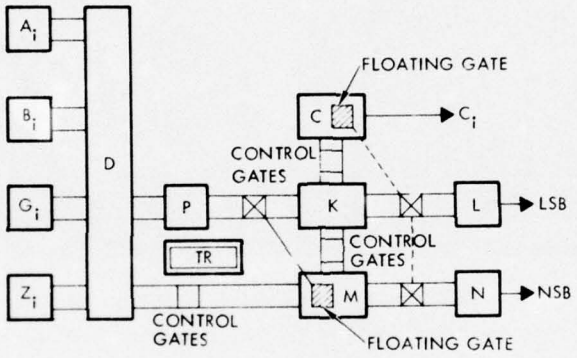


Figure 2-17. Four-Input Adder

The two-input adder performs in the following way: the shift registers A_i and B_i simultaneously directly dump their charge packets into the potential well under gate D. The potential wells under gates D, C, L, and N are all equal to each other.

During the time interval when the shift registers A_i and B_i are dumping charge into D, transfer gate T_S is biased off, allowing no direct charge flow from D to L.

The control gate, during the above dumping interval, allows surplus charge to flow under the floating gate master side, gate C. The floating gate master side detects the presence or absence of charges under itself and allows charge under gate D to flow to gate L only if there is no charge under gate C. This is achieved by controlling the surface potential under gate C, when the transfer gate, T_S, is biased on.

The outputs of gates L and N represent the least significant bit and the next significant bit respectively of the sum. The purpose of the transfer gate (TR), is to clear out any excess charge left under gate D before the start of the next sum sequence.

The uniqueness of the two-input CCD adder is in the ease of design and its ability to add two words, 2 bits at a time without many of the time delays inherent in other CCD adders.

Four-Input Adders

As indicated in Figure 2-17 the four-input full adder differs from the original three-input full adder configuration, by the addition of the shift register Zi and gates C, L, and N. Basically, the circuit performs in the following way. The shift registers Ai, Bi, Gi, and Zi all simultaneously dump their charge packets into the potential well under gate D. As in the basic configuration, potential wells under gates A, B, G, Z, D, M, K, and C are all equal to each other. This, in conjunction with the control gates indicated in Figure 2-17, allows surplus charge to flow in a sequential manner from under gates D, M, K, and C respectively. Overflow charge cannot flow directly from the potential well under gate D to the potential well under gate K, because gate P is biased in the off position during the overflow interval. That is, during the time interval when the shift registers Ai, Bi, Gi, and Zi are dumping charge into D, gate P is biased off, allowing no direct charge flow.

The charge sensing element under gate M will allow charge under gate to flow to gate K during the time interval immediately after the dumping interval only if there is no charge under gate M. This also is the case for charge flow from gates K and M to gates L and N respectively. That is, the charge sensing element under gate C will allow charge under gates K and M to flow the gates L and N respectively, only if there is no charge under gate C.

The output of gate C is the look ahead carry and is represented by Ci. The outputs of gates L and N are the least significant bit and the next significant bit respectively of the sum.

The purpose of the transfer gate (TR), is to clear out any excess charge left under gates D, K, and M before the start of the next sum sequence.

Test Results

As indicated in paragraph 2.2.1 the DP-0 mask differs from the LSM-1 mask in that a FET discharge circuit was added to allow discharge of the floating gate structure. Tests performed on the floating gate amplifier devices and three-input adders

from the DP-0 mask set concentrated on devices which had the reset FET connected to the floating gate structures. The tests were to determine:

- Effective resistance across the floating gate resulting from the reset FET drain diffusion
- Conformance of these devices to previous models.
- Successful operation of the three-input full adder.

Test of the DP-0-1, 2,3, and 5 lots indicated a very poor yield of full adder circuits. The poor yield was due to metal line breakage over poly steps. This has been corrected in the DP-1 mask. Table 2-4 summarizes some of the DP-0 results. Test measurements revealed that the equations which model the operation of the floating gate amplifier, as designed on DP-0, will have to be modified to reflect the floating gate amplifier discharge configuration

Tests also indicated that the addition of the FET discharge to the floating gate structure ensures the floating gate performance to be nonsusceptible to charge accumulation on the gate. This FET (as predicted) has contributed to the successful operation of the three-input full adder as evidenced by Figures 2-18, 2-19, and 2-20.

Once the proper setup conditions were made to the three-input full adder, and without any further adjustments, the following characteristics were observed:

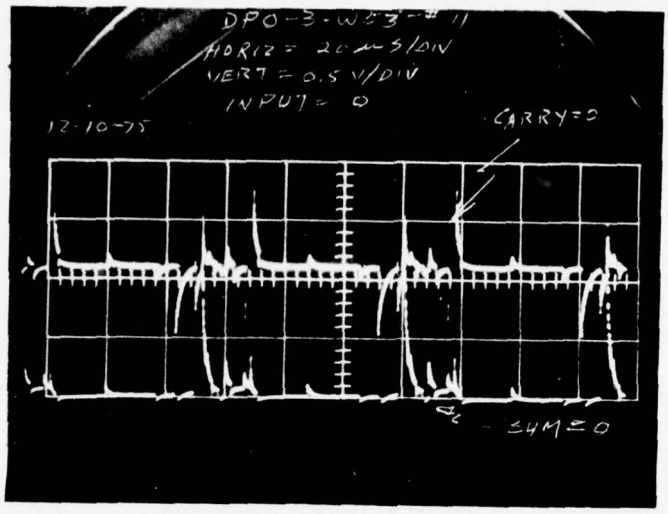
- When one input is applied to the adder a sum signal output is observed (Figure 2-18)
- When two inputs are applied to the adder a zero signal and a one signal are observed in the sum and carry outputs respectively (Figure 2-19)
- When three inputs are applied to the adder, one signal is observed in both the sum and carry outputs (Figure 2-20).

The above observed results are precisely the outputs expected when the above respective signals are applied to the full adder input.

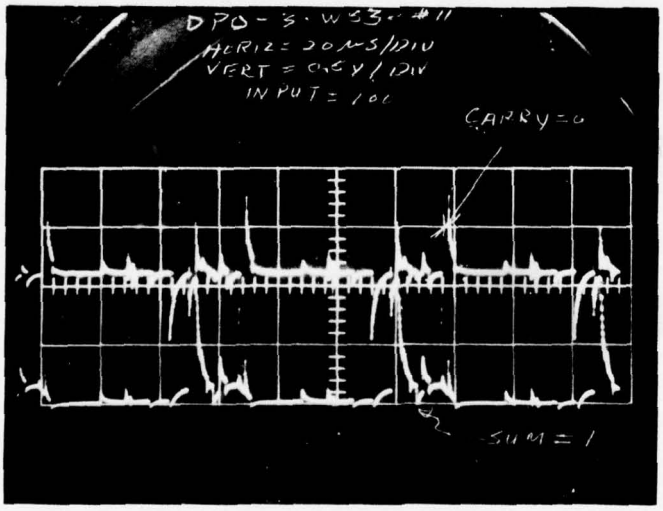
As theoretically predicted, the sum output for a single one input does not exhibit the output magnitude of the other two cases. This variance in magnitude is due to the geometrical design and layout of the floating gate structure incorporating the FET discharge capability on DP-0; the DP-1 design will produce equal amplitude output voltages for all cases.

Table 2-4. Summary of Lots DPO-1, -2, -3, -4, and -5

Lot No.	Major Processing Characteristics						Device/Circuit Characteristics						
	Poly/Oxide Thickness		Metal/Oxide Thickness		Field Oxide		Test FETs (Threshold Voltages)				Floating Gate Amplifiers	Full Adders	Remarks
	Wet	Dry	Wet	Dry	Wet	Dry	Poly/ Gate	Metal/ Gate	Poly/ Field	Metal/ Field			
DPO-1 Wafer 40 41 42	1000Å		2000Å		X		-7.0	-9.0	-35.0	-27.0	V_S and V_G curve can't repeat	Light sensitive test Wafer 40 Sum channel Good: 14 Bad: 12 Carry channel Good: 20 Bad: 6	Shift register result Wafer 40 Good: 9 Bad: 17 Wafer 41 Good: 5 Bad: 21 Wafer 42 Good: 6 Bad: 20
												Wafer 41 Sum channel Good: 16 Bad: 10 Carry channel Good: 24 Bad: 2	Full adder output circuit tested by light very good but operating as shift register resulted in low yield Some good units functional as full adder
												Wafer 42 Sum channel Good: 23 Bad: 3 Carry channel Good: 23 Bad: 3	
DPO-2 Wafer 51 52		1000Å		2000Å	X		Bad	--	--	--	Bad	Bad	All FET – floating gate and full adder were in depletion mode
DPO-3 Wafer 53 54 55		1000Å	2000Å		X		-1.5	-2.0	-16.5	-8.5	V_S and V_G curve ok	Light sensitive test Wafer 53 Sum channel Good: 16 Bad: 6 Carry channel Good: 21 Bad: 1	Full adder functional with +10 V substrate. Bias output FET circuit performance not affected by increased frequency from 28 kHz to 71 kHz. Operate as full adder: Input = 111; sum output decreased as frequency increased
												Wafer 54 Sum channel Good: 9 Bad: 13	Light test result indicate sum channel output low
												Wafer 55 Sum channel Good: 4 Bad: 18 Carry channel Good: 19 Bad: 3	Shift register test result indicates low yield on good unit
												Shift register test Wafer 53 Good: 5 Bad: 22	
												Wafer 54 Good: 8 Bad: 18	
												Wafer 55 Good: 7 Bad: 19	
DPO-5 Wafer 57 59	1000Å		2000Å		X		-3.0	-4.5	-35.0	-22.0	*Omitted testing*	Light sensitive test Wafer 57 Sum channel Good: 4 Bad: 18 Carry channel Good: 20 Bad: 2	Light test on both wafers. Each had 3 good units and all 3 good units on same location at each wafer Shift register test resulted in low yield
												Wafer 59 Sum channel Good: 3 Bad: 19 Carry channel Good: 19 Bad: 3	
												Shift register test Wafer 57 Good: 3 Bad: 24	
												Wafer 59 Good: 3 Bad: 24	



a) NO INPUT APPLIED



b) SINGLE "ONE" INPUT APPLIED

Figure 2-18. Output to Three-Input Adder

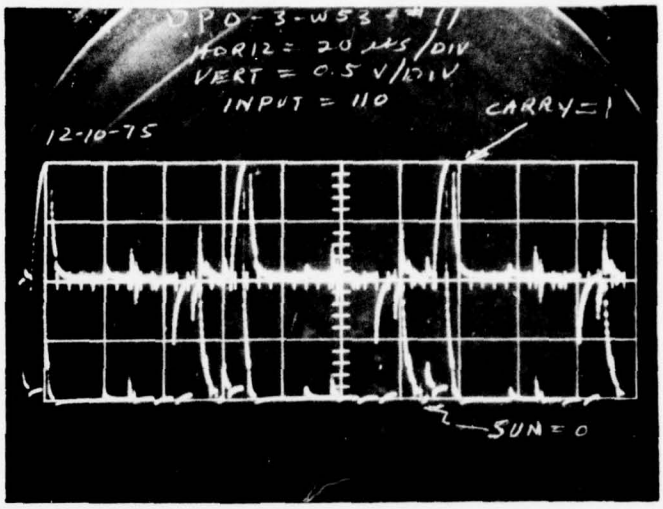


Figure 2-19. Sum and Carry Output When Two "Ones" Are Applied to Three-Input Adder

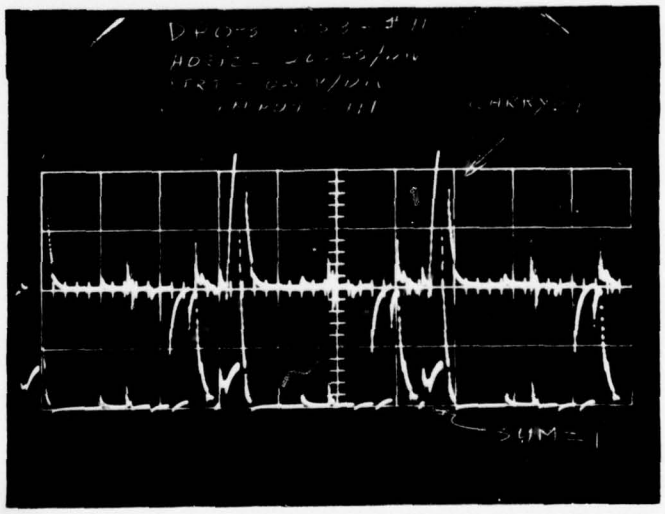


Figure 2-20. Output with Three "Ones" Applied to Three-Input Adder

2.3.4 Three-Input Full-Adder Designs on Mask Set DP-1

The logic design of multiplier or adder arrays can be mechanized from half-adders, full-adders or a mixture of both. Design tradeoffs are associated with all three approaches; the half-adder approach requires more CCD gates than if only full-adders are used but has a simpler clock-line layout. A mixture of half-adders and full-adders requires the least number of CCD gates but it also requires the design of two standard logic functions.

In this section we describe the design of a full-adder logic function, its modification to half-adder and the design of a two-input AND gate. The designs of a two-word adder array and a multiplier array are also described.

A full-adder logic cell has three inputs: a, b, and g, of which any one may have a binary value of 0 or 1. The logic cell adds the three bits together and generates a sum bit and a carry bit, each having a binary value of 0 or 1. A truth table for the full-adder logic cell is given in Table 2-5.

Table 2-5. Truth Tables for Full-Adder, Half-Adder and AND Gate Logic Functions

Inputs			Output	
a	b	g	c	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(a) Full-adder truth table

Inputs		Outputs	
a	b	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(b) Half-adder truth table

Inputs		Outputs
a	b	z
0	0	0
0	1	0
1	0	0
1	1	1

(c) AND gate truth table

The operation of the full-adder cell can best be described by referring to the schematic shown in Figure 2-21. In the design the charge storage buckets are the CCD channels under the polysilicon and are labeled in Figure 2-21 as A, B, C, etc. If any of the three input signals a, b, g are at the binary 1 level they allow the corresponding input buckets A, B, G to fill. When the Ø1 clock line goes to its negative value, the charges in A, B, G transfer to the D storage area. Since the D storage area is identical in size to A, B, or G it will fill completely if any one of the three input buckets is full. However, if two input buckets are full, it will overflow and also completely fill storage area C under the floating gate. The charge in the C storage area will cause a voltage change to occur on the master end of the floating gate. This voltage change is immediately transferred to the slave end, which in turn causes a change in the surface potential of the CCD channel beneath it, inhibiting the transfer of charges along the channel.

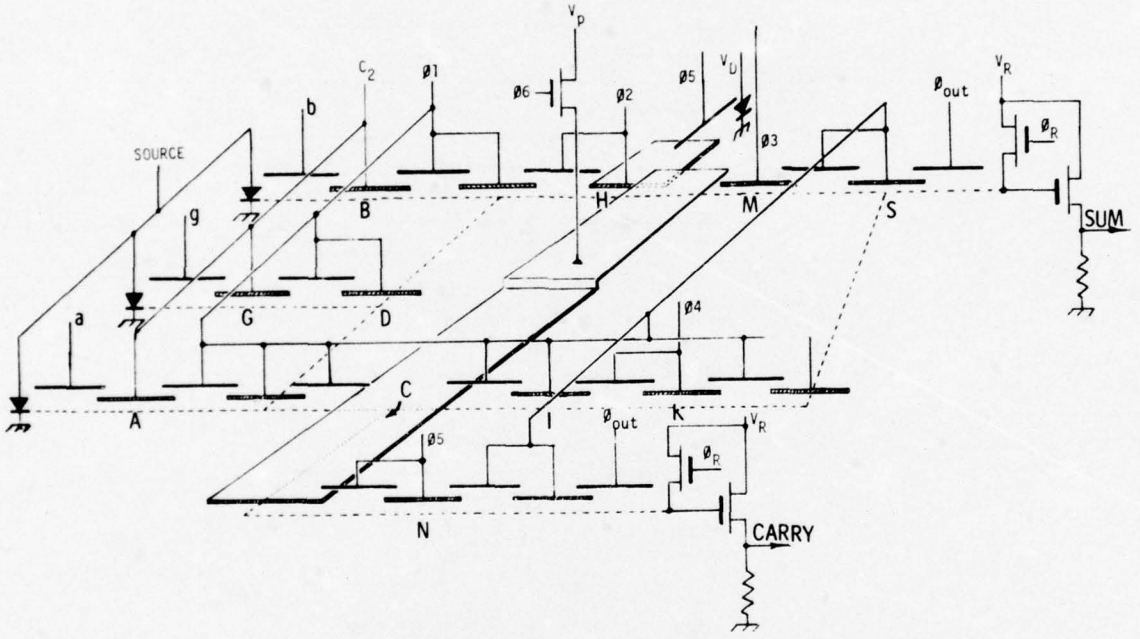


Figure 2-21. Schematic Diagram of Full Adder Test Cell

When all three input buckets A, B, G are full, then both the D and C storage areas spill over and the I storage area also fills completely.

For the first condition described, when only one input bucket is full, the charge stored in D is transferred into the H storage area, then under the slave end of the floating gate into M, and subsequently into S at the next Ø1 negative transition and out as a binary 1 sum bit.

When two input buckets fill both the D and C storage areas, the charge in D again transfers into H, but now it is inhibited from passing under the floating gate. It is

removed from the H storage cell before the next input bit appears by Ø5 going negative and transferring the charge to the discharge diode; and the SUM bit is now an 0. Also at the Ø5 negative edge, the charge stored in the C storage area is transferred out and becomes a binary 1 carry bit. For the final condition when all three input buckets and also the D, C, and I storage buckets are full, the identical transfer conditions exist as when two input buckets are full; except that the charge in the I storage area transfers out at the Ø4 negative transition. Thus both the sum and carry outputs have a binary 1 output.

The sequences of charge storage and transfer can be derived from the waveforms of the full-adder clock phases shown in Figure 2-22.

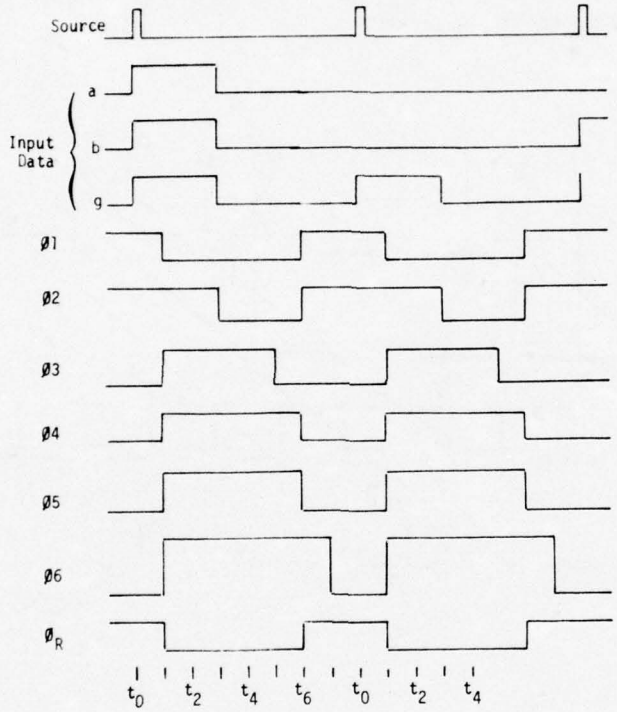


Figure 2-22. Clock Line Phase Sequence for Full-Adder and Half-Adder Functions

Several clock lines are required for the full-adder cell, and in an attempt to keep the clock lines to a minimum number, it was decided to make the storage areas of A, B, G, D, M, I, S, K, N identical. For the layout reasons explained in Section 4.3 the standard area of these standard gates is 6.9 mil^2. In each of the logic cells described the aluminum gate is used for transfer and the polysilicon gate is used for storage. In general, an aluminum gate is connected to a polysilicon gate. The maximum amount of charge that can be stored in the D, M, I, etc., storage areas can be represented by Q_s;

$$Q_s = A_s[K_s(\sqrt{\phi_p} - \sqrt{\phi_a}) + C_{ox}\ (\phi_p - \phi_a)] \tag{2.8}$$

where K_s is a constant that involves the semiconductor processing parameters, A_s is the surface area of the gate C_{ox} is the oxide capacitance per unit area, A_p is the value of surface potential under the polysilicon gate, and ϕ_a is the value of surface potential under the metal gate. The surface potential is measured at a specific gate voltage and obtained from the curves plotted from a wafer that was processed to an identical schedule gate voltage-surface potential used for the current design.

During the design of DP-1, only as the first term of equation (2.8) was retained as an approximation to the actual Q_s. Thus for the DP-1 process, at a Ø1 gate voltage of -8.7 volts $\phi_p = 3.5$ and $\phi_a = 1.25$

$$Q_s = K_s\ 0.69\ (\sqrt{3.5} - \sqrt{1.25})$$

$$= K_s\ 0.518$$

In order to obtain as large voltage change under the master end of the floating gate, the C storage area was made as small as possible. The minimum area was controlled by the spacing of three metal gates that overlap the C polysilicon area, and resulting in an area of $A_c = 0.3\ \text{mil}^2$.

When two input signal lines are at binary 1, both the D and C storage areas must fill completely, yet not spill over to the I area. Therefore the full level of the C bucket must be identical to that of the D area. If the approximate well size expression is rearranged, the empty well potential, ϕ_e, under the floating gate can be derived

$$\phi_e = \left(\frac{Q_s}{K_s A_c} + \sqrt{\phi_a}\right)^2 \tag{2.9}$$

$$= \left(\frac{0.518\ K_s}{K_s\ 0.3} + \sqrt{1.25}\right)^2$$

$$= 8.09$$

However, positive feedback is induced to the C storage area from the floating gate. As the C storage area fills with charge, the induced voltage change on the floating gate results in an apparent reduction of the empty C bucket size. Since a full bucket depth of $\phi_e - \phi_a$ is required it is necessary to precharge the floating gate to an initial voltage that compensates for the feedback reduction. The surface value ϕ_i, of the required empty bucket under the floating gate can be derived from

$$\phi_i = (\phi_e - \phi_a) \frac{C_{ox-1}}{C_p} + \phi_e \qquad (2.10)$$

where

C_{ox-1} = oxide capacitance over the C storage area

C_p = total parasitic capacitance of the floating gate

For the subject gate design C_{ox-1} = 0.034 pf and C_p = 0.058 pf

$$\phi_i = (8.09 - 1.25)\ 0.592 + 8.09$$

$$= 12.1 \text{ volts}$$

From the gate voltage-surface potential curves for polysilicon, a gate voltage of V_G = 17.8 volts is required to produce a surface potential V_s = 12.1 volts.

At the appropriate time the charge in the C storage area is transferred into the N storage area by the negative transition of Ø5. In order to transfer all of the charge in the C storage area, it is necessary to make the Ø5 negative level be -24 V which results in a 0.9 volt step from the ϕ_i level of 12.1 down to 13 volts.

The full-adder and the half-adder cells are designed to have a hybrid floating-gate; the master end is polysilicon and the slave end is aluminum. The 17.8 precharge voltage applied to the floating gate results in an initial V_s under the (aluminum) slave end of ϕ_{s1} = -9.4 volts.

When the C storage bucket is filled, it induces a voltage change in the floating gate resulting in a new value V_g at the slave end

$$V_g = V_g - (\phi_i - \phi_a) \left(\frac{C_{ox-1}}{C_{ox-1} + C_p} \right) \qquad (2.11)$$

$$= 17.8 - (12.1 - 1.25)\ (0.37)$$

$$= 13.8 \text{ volts}$$

The 13.8 volts on the metal end of the floating gate produces a new ϕ_{s2} = -6.5 volts. The transfer gate under the slave end of the floating gate then switches from -9.4 to -6.5 volts.

To ensure that the charge stored under the H bucket is completely transferred when the floating gate induces the ϕ_{s1} level, its empty level ϕ_{he} must be approximately one-half volt less negative than ϕ_{s1}

$$\phi_{he} = \phi_{s_1} + 0.5$$

$$= 9.4 + 0.5$$

$$= -8.9 \text{ volts}$$

However, to ensure that the ϕ_{s_2} level under the floating gate also completely inhibits transfer of the H bucket charge, the full bucket level, ϕ_{hf}, in the storage area must not exceed ϕ_{s_2} - 0.7 volt.

$$\phi_{h_f} = \phi_{s_2} - 0.7$$

$$= -6.5 - 0.7$$

$$= -7.2 \text{ volts}$$

The area of the H storage area can be derived by rearranging the approximate well bucket size expression

$$A_H = \frac{Q_s}{K_s \left(\sqrt{\phi_{h_f}} - \sqrt{\phi_{he}}\right)} \tag{2.12}$$

$$= \frac{K_s \; 0.518}{K_s \left(\sqrt{8.9} - \sqrt{7.2}\right)}$$

$$= 1.8 \text{ mil}^2$$

During the enable mode the charge stored in the H area is transferred under the slave end of the floating gate into the M storage area. To ensure that all of the charge is transferred it is necessary that the full well level ϕ_m, of the M storage area be approximately one volt more negative than ϕ_{s_1}

$$\phi_m = \phi_{s_1} - (\phi_p - \phi_a) - 1.0$$

$$= -9.4 - (3.5 - 1.25) - 1.0$$

$$= -12.65$$

The voltage applied to the ϕ_3 polysilicon gate necessary to produce ϕ_m = -12.65 is -18.5 volts. The design of the half-adder logic cell is very similar to the design of the full-adder, and a schematic diagram of the half-adder is shown in Figure 2-23. Since there are only two inputs (a, b) to a half-adder the need for the I and K storage cells disappears, so the layout is simpler.

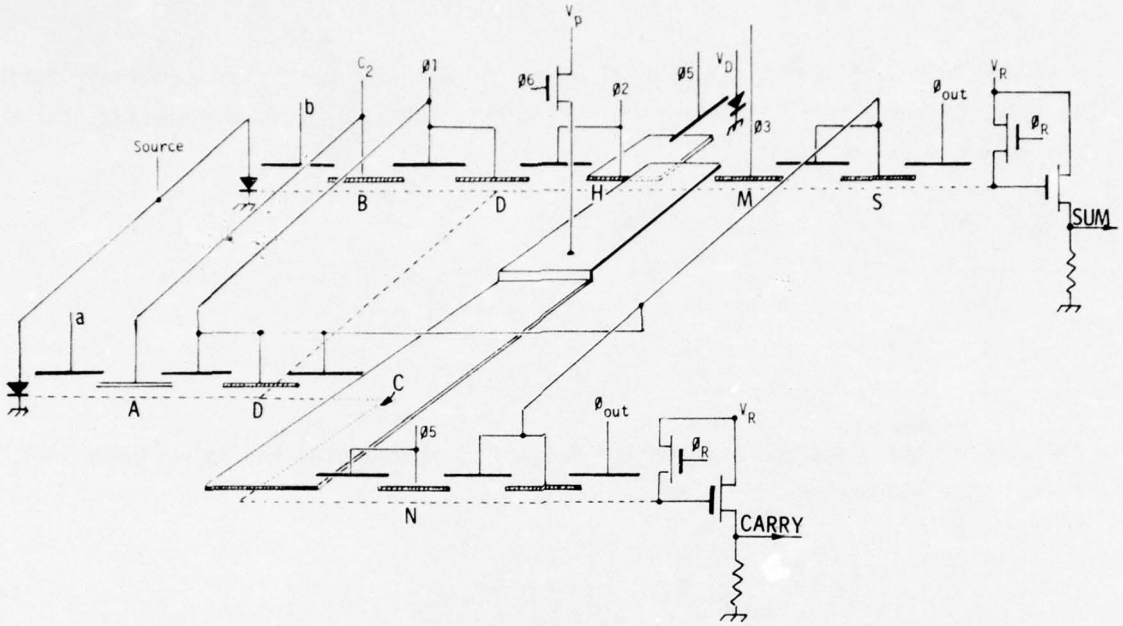


Figure 2-23. Schematic Diagram of the Half-Adder Test Cell

A truth table for the half-adder logic cell is given in Table 2-5; the clock sequences shown in Figure 2-22 are also applicable to the half-adder, except that ø4 is not required.

To perform binary multiplication it is necessary to use a two-input AND gate. This function was mechanized by removing the floating gate and the H, M, and S storage areas from the half-adder design. It was also necessary to move the discharge gate and diode from the H storage area to the D storage area, as shown in Figure 2-24. A truth table for the AND gate is given in Table 2-5. The AND gate operates from only two clock phases.

In order to obtain design, layout, and testing experience with CCD signal processing devices, a two word 4-bit binary adder was included on the DP-1 chip. The addition of the two binary words a_1 - a_4, b_1 - b_4 is performed in a straightforward manner:

Carry bit	C_4	C_3	C_2	
First word	a_4	a_3	a_2	a_1
Second Word	b_4	b_3	b_2	b_1
Sum	C_5	s_4	s_3	s_2	s_1

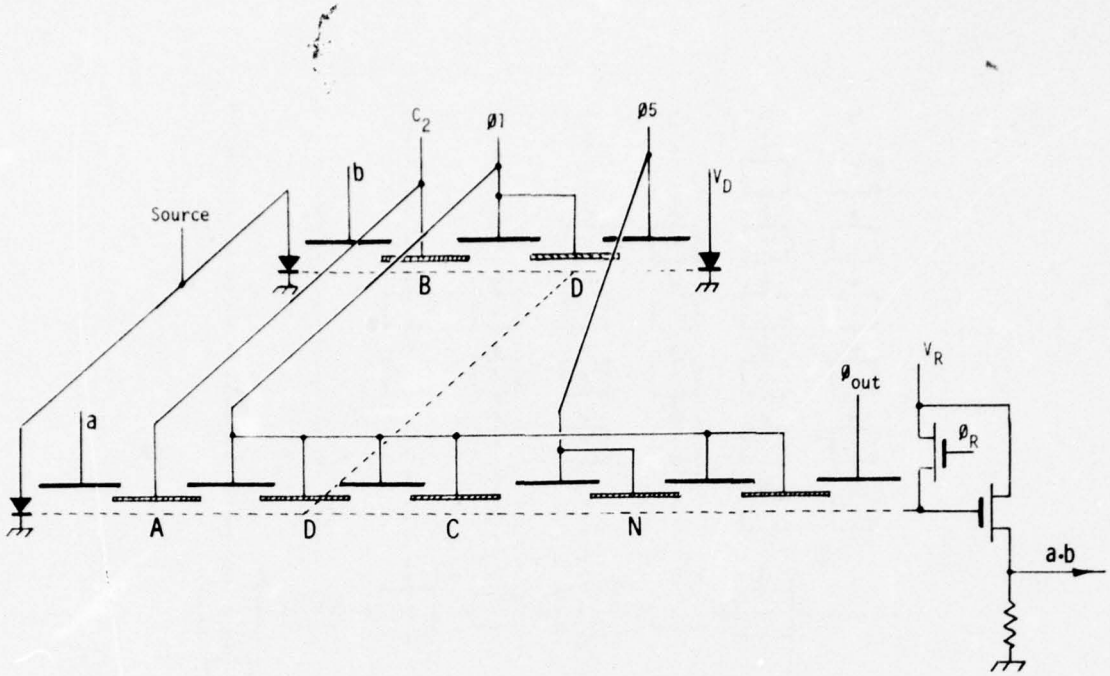


Figure 2-24. Schematic Diagram of Two-Input AND Gate

The least significant column has only two input bits so that a half-adder is satisfactory; however, the other three columns have three inputs and full-adders are required.

A block diagram of the two word, 4-bit adder is shown in Figure 2-25. It will be seen from the diagram that delay stages have been added to the input signal paths of the most significant bits; this is to ensure that the input data arrive at the full-adder output synchronously with the carry bit. In order to compensate for the input delays applied to the most significant bits, it was also necessary to include corresponding delays to the sum output lines of the least significant output bits.

A two word 3-bit multiplier array was also included on the DP-1 chip. The multiplier is a little more complex than the adder and involves the use of an AND function besides half-adders and full-adders. The multiplication of the two 3-bit numbers $a_1 - a_3$, $b_1 - b_3$ is performed in the usual manner

			a_3	a_2	a_1
			b_3	b_2	b_1
			a_3b_1	a_2b_1	a_1b_1
		a_3b_2	a_2b_2	a_1b_2	
	a_3b_3	a_2b_3	a_1b_3		
P6	P5	P4	P3	P2	P1

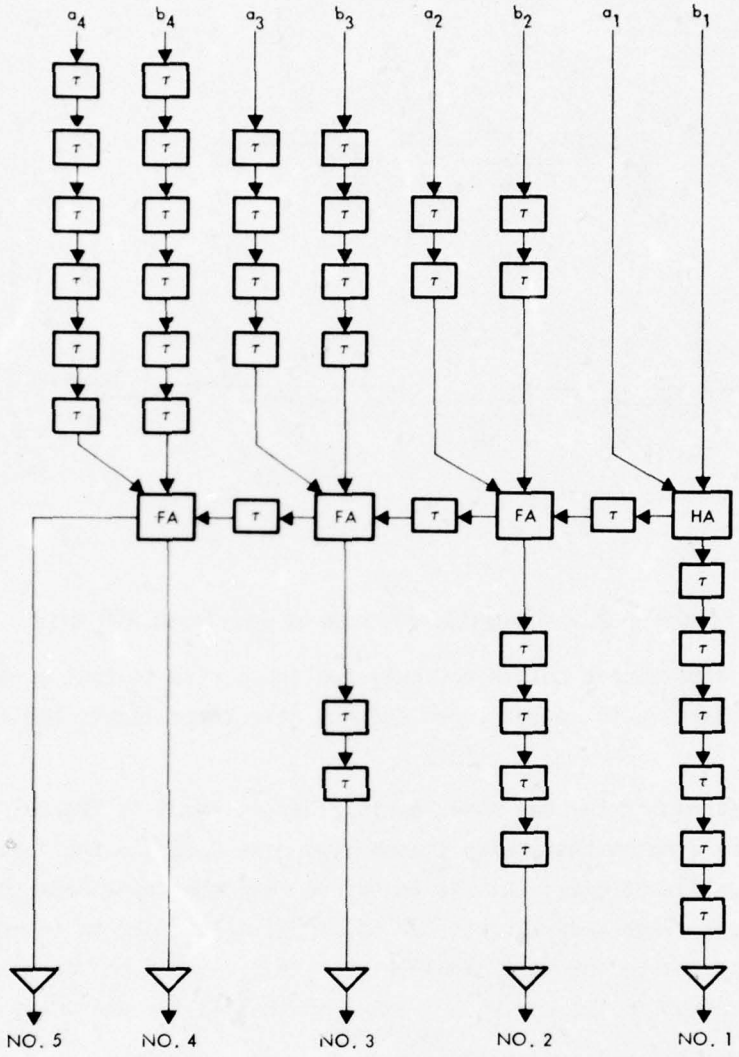


Figure 2-25. Two-Word 4-Bit Adder Using Three-Input Adders

The least significant column consists of an AND gate, the second least significant column requires two AND gates and a half-adder. The third column requires three AND gates; and a carry bit may also be received from the second least significant column and must be added to the three output bits from the AND gates. Thus, overall, the mechanization of that column used a full-added and a half-adder as shown in the block diagram of the 3x3 multiplier (Figure 2-26). In the fourth column from the right, two AND gates are required; however, since a carry bit may be received from both the half-adder and full-adder of the lower column, again four bits must be added together. An identical combination of full-adder and half-adder were used to add the four bits. The second most significant column requires only an AND gate and full-adder, the carry output from the full-adder providing the most significant bit in the product.

In order that the product bits were obtained in phase, it was necessary to add delay stages to the most significant input and least significant output columns. These delays are shown in Figure 2-26.

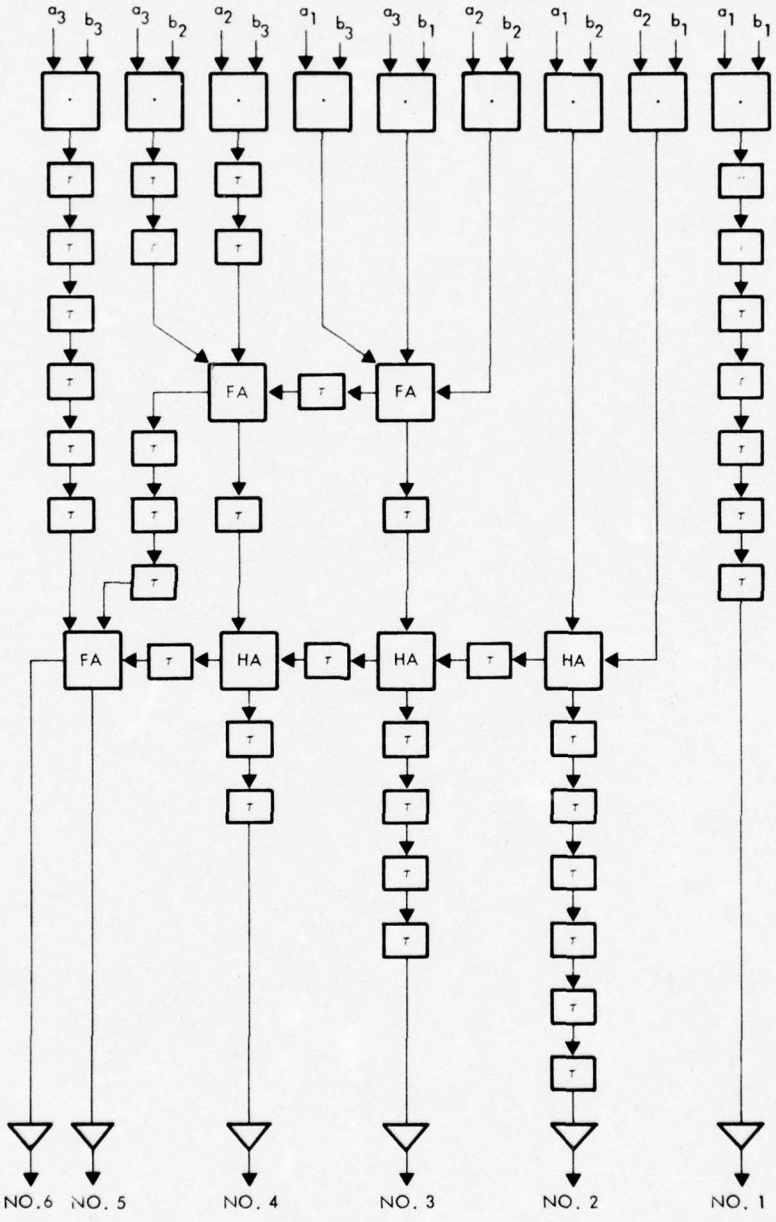


Figure 2-26. 3 x 3 Pipeline Multiplier

3. COMPUTATIONAL ALGORITHM TRADEOFFS

Use of the two, three, and four-input adders to develop and design digital systems, such as N-bit multipliers and adders, revealed interesting system configurations and design requirements.

A few conceptual systems which utilize the two and three-input adder and other CCD circuit designs are depicted in Figures 3-1 through 3-4. The block diagrams indicated in Figures 3-2 represent the adder and multiplier designs of the DP-1 mask set.

The two-word 4-bit adder using two input adders and OR gates as the basic building blocks can be implemented in many ways, two of which are depicted in Figures 3-1 and 3-2.

Both designs accept the incoming information directly into the two-input adder. The adder (Figure 3-2) operates on the concept that the sum and carry outputs of the adjacent 1/2 adder are the inputs to succeeding 1/2 adders. Since the data generated by adjacent adders are generated in a parallel fashion, no delays are inherent in the adding operation of the array. The appropriate sum outputs are delayed to keep the information moving synchronously through the adder.

Since the output of the two-input adder is either a (01) or (10) representing the inputs of a (1,0) or a (1,1) respectively, OR gates are used to truncate the array design in lieu of input adders. Additional features of the OR functions are design simplicity, operation, and minimal real estate requirements as compared to the two-input adder.

The adder array indicated in Figure 3-1 utilizes the OR function within the array between successive two-input adders. This allows simplicity of design and utilizes less two-input adders, as compared to the two-input adder array discussed above. As an example, the two-input adder array (Figure 3-2) requires 10 two-input adders and three OR gates to perform the same arithmetic function as the two-input adder (Figure 3-1) which requires only seven two-input adders and three OR gates. Twenty-four delay gates are required by the latter design as compared to 11 delay gates required by the former design. Even with the additional 13 delay gates the two-input adder array will require less real estate than the design of Figure 3-2, since delay gate real estate requirements are very small.

The two additional delays required in the design (Figure 3-1) compare to the adder (Figure 3-2) needed to maintain synchronization of all the data lots will degrade the signal level due to transfer inefficiency. Optimum design, therefore, is to minimize the amount of delayed transfers required to perform arithmetic functions.

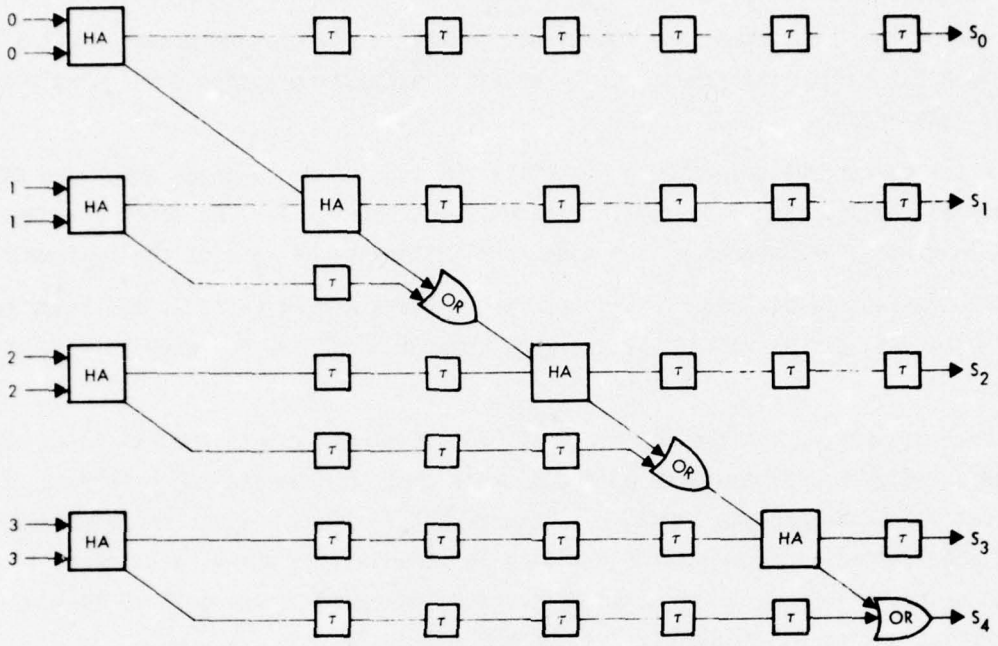


Figure 3-1. Four-Bit Full Adder Using Half Adders

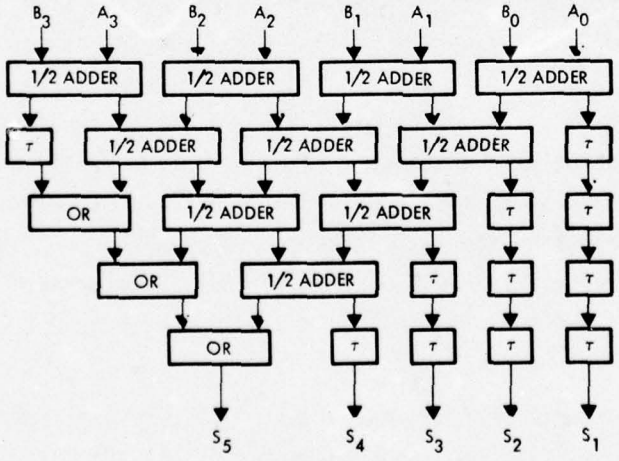


Figure 3-2. Two-Word 4-Bit Adder Using Two-Input Adder Circuits

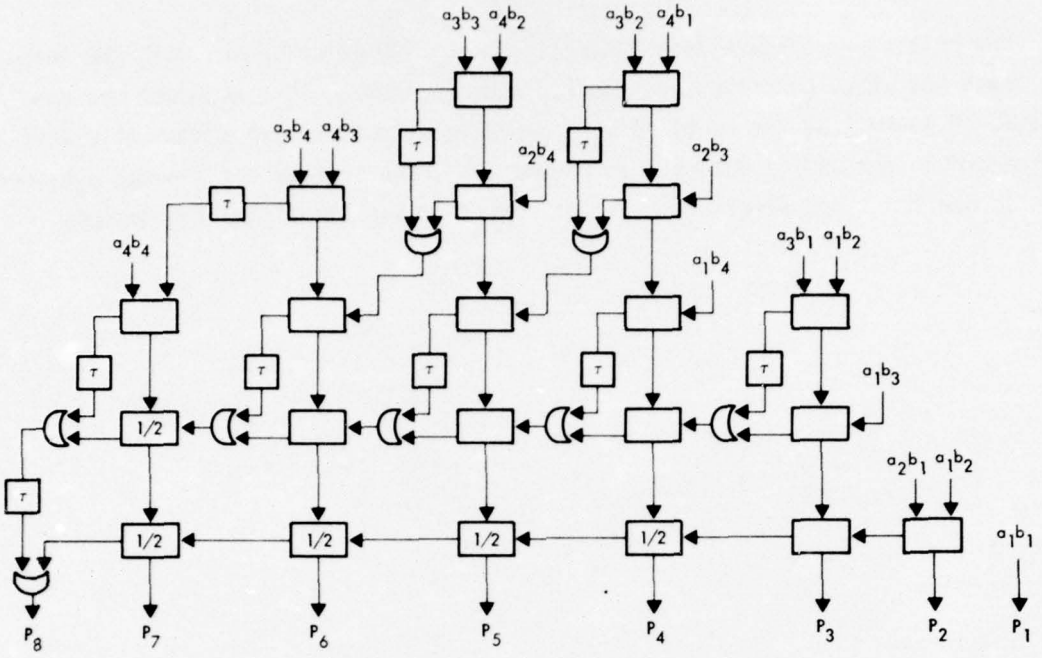


Figure 3-3. 4 X 4 Multiplier Using Half (2,2) Adders

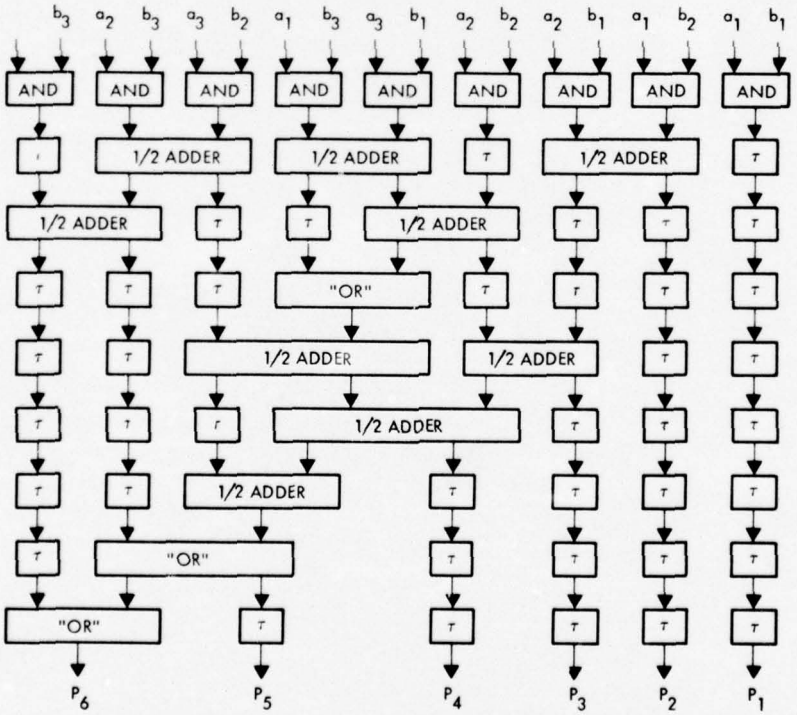


Figure 3-4. Two-Word 3-Bit Multiplier Using Two Input Adders and OR Circuits

The multiplier (Figure 3-4) basically is the two-input adder, with AND gates in the front end which generate the required $a_i b_i$ products. This approach was used instead of generating the required $a_i b_i$ products at the various times and places they are needed in the design depicted in Figure 3-3. The approach of product $a_i b_i$ generation in the front end eliminates channel crossings and simplifies the design.

4. MASK SET DP-1

4.1 OVERVIEW

The mask set DP-1 design was based in part on the experience gained through the use of mask set DP-0. As the concepts for arithmetic computation continued to evolve, two basic schemes emerged. Previous chapters discussed some of the important trade-offs involved in comparing these approaches. One purpose of DP-1 was to permit additional experimental investigation of these techniques. Accordingly, DP-1 contained two realizations of each circuit function; one realization employed three input adders, the other two input adders. The list of circuits and test devices on DP-1 is contained in Table 4-1. Chips produced from this mask set provide complete test vehicles for comparing and contrasting the two diferent approaches and for gaining experience with arithmetic function design, production, and testing. Figure 4-1 shows an overall view of a DP-1 chip.

Table 4-1. Circuits and Test Devices on DP-1

Circuits
Two-word 3-bit multiplier
Two-word 4-bit adder
Test Devices
AND gates
Two-input adder
Three-input adder
Threshold test FETS

4.2 DETAIL DESCRIPTION OF TWO-INPUT FULL ADDER

The general operation of the two-input adder is discussed in Section 2.3.3 of this report. A photograph of the circuit is depicted in Figure 4-2. Figure 4-3 depicts operation of the two-input adder as a function of the surface potentials, gate voltages, and timing waveforms. The equations used in the two-input design, in determining the gate voltage, charge, and respective surface potentials are presented in Section 4.2.2.

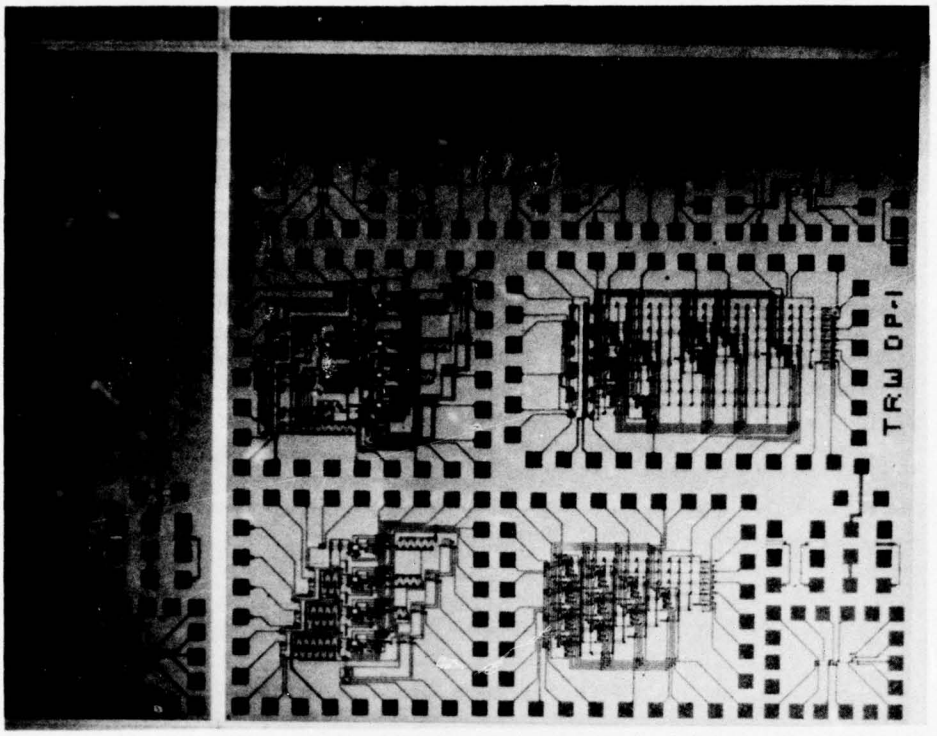


Figure 4-1. DP-1 Chip

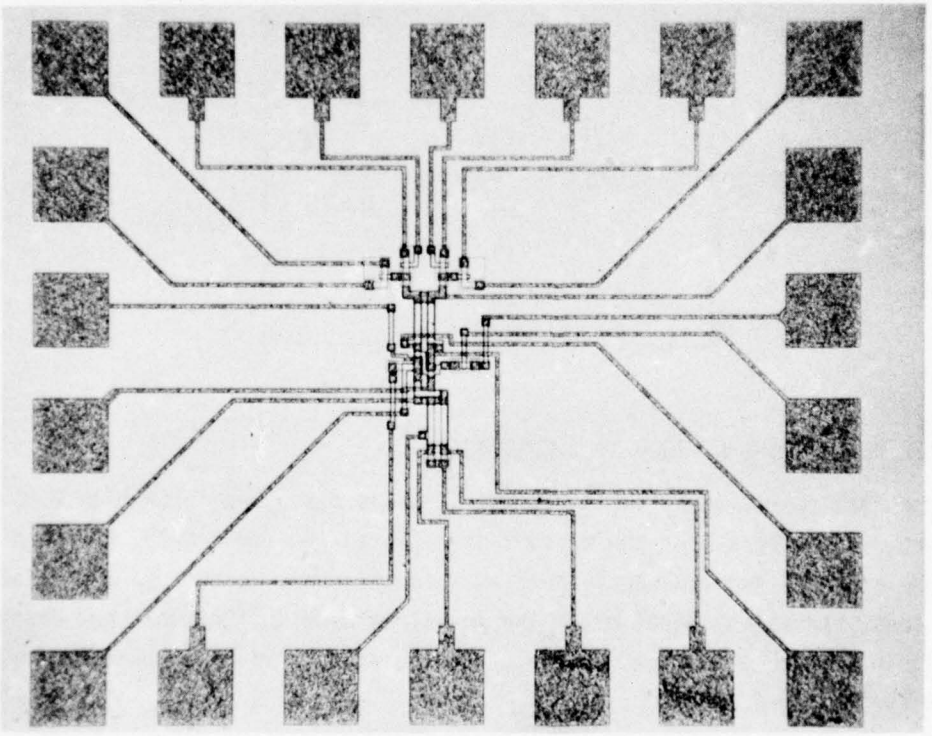


Figure 4-2. Two-Input Adder Circuit

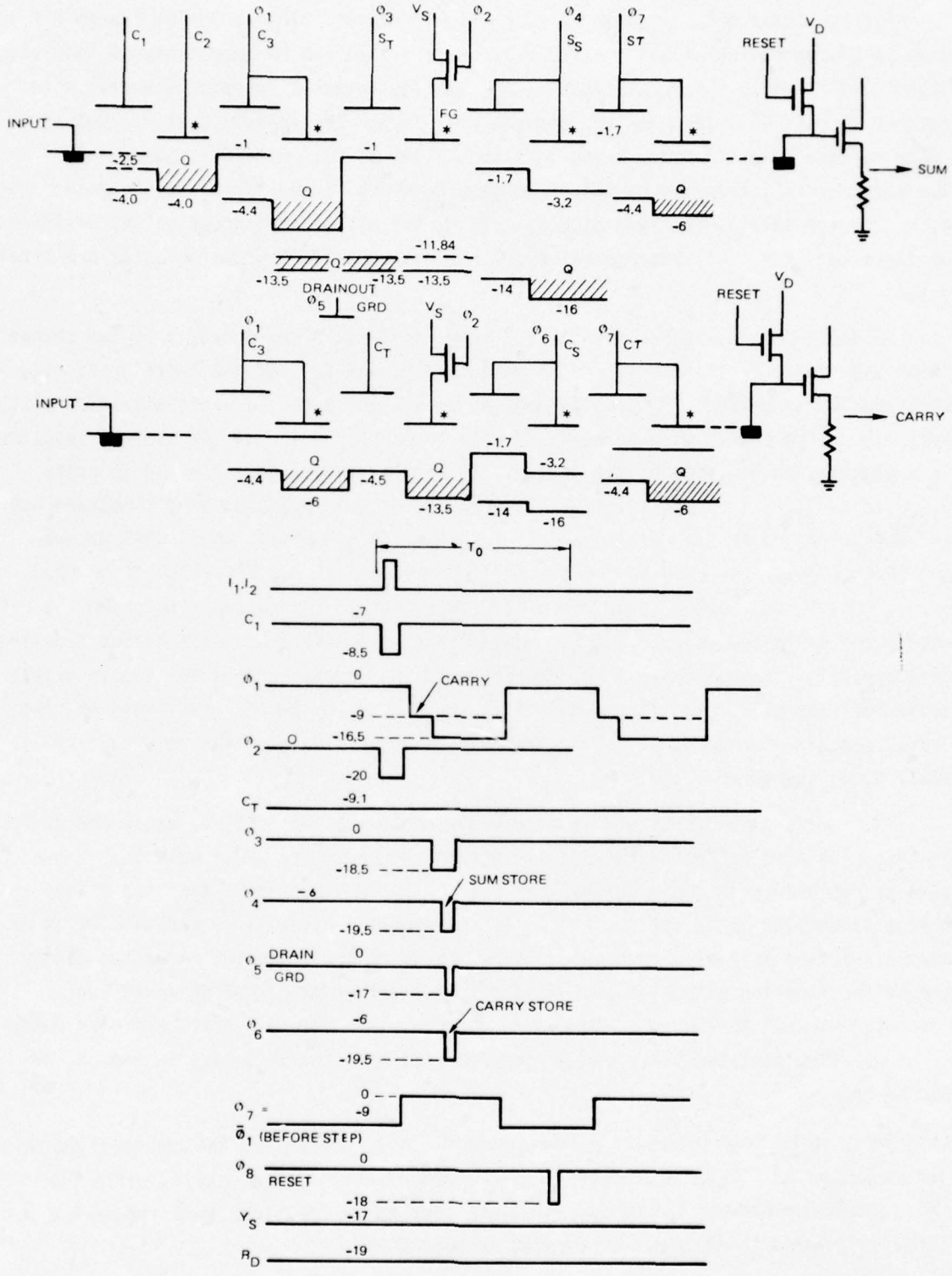


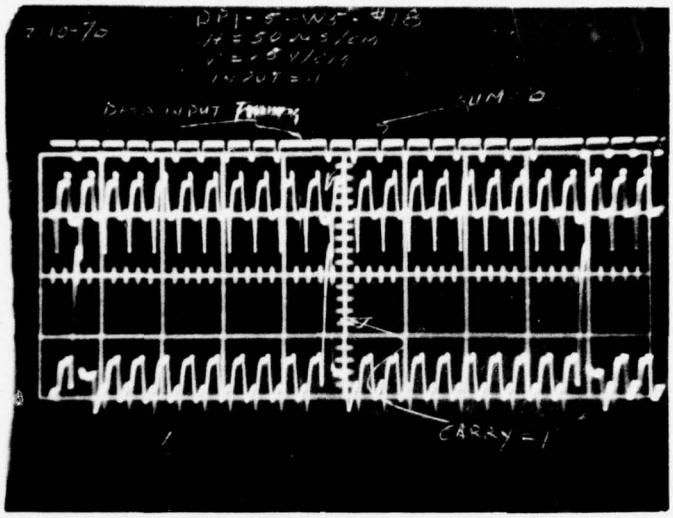
Figure 4-3. Detail Schematic Indicating Surface Potentials, Gate Voltages, and Timing Waveforms of Two-Input Adder

The operation of the two-input adder is as follows. The amount of charge Q to be used in the operation of the two-input adder is determined by input gates C_1 and C_2. Gate C_1 is actually the input signal gate, but its relative surface potential with respect to gate C_2 determines the storage well size. The input diodes are pulsed on, their surface potential then being just above that of C_1. Gate C_3 is used as a transfer gate and is internally connected to gate S_1 which is the first storage gate. The first storage gate is the same size as gate C_2 and all other storage gates, designated by the asterisk *. All storage gates are poly gates and all transfer gates are metal gates.

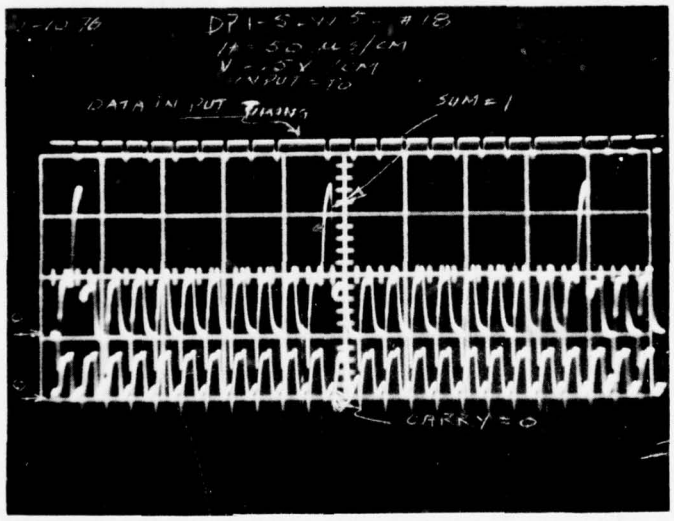
As indicated in Figures 4-2 and 4-3, the two-input adder consists of two channels: "sum" and "carry." Their only common gates are C_3 and S_1. If the incoming signal, when applied to gates $C_{1a,b}$ simultaneously, is a (1,0) or (0,1), this signifies that only one charge packet is generated and will be processed in the sum channel resulting in a digital sum of (1,0) at the output. If the incoming signal applied to gates $C_{1a,b}$ is a (1,1), this signifies that two charge packets are generated simultaneously and are processed in the carry channel, resulting in a digital sum of (01) at the output (Figure 4-4). As seen by the timing diagrams, when input signal (1,0) or (0,1) occurs (C_T) in the carry channel is biased such that no charge can flow under the gate designated V_S in the carry channel. This gate is the master end gate of the floating gate amplifier. Since there is no charge under the master gate of the floating gate amplifier, the slave end is biased such that when gate S_T and S_s are turned on, the charge contained under S_1 will flow from S_1 and be stored under the poly gate of S_s until S_T is turned on.

If, however, the input signal is a (1,1), this implies that two equal charge packets have been generated and are simultaneously processed under gate S_1. Since gate S_1 can only hold one Q quantity of charge, C_T is biased such that the other charge is dumpled under the master end of the floating gate, since the gate S_T is biased off during this time interval. Since charge is now deposited under the master end of the floating gate, the slave end of the floating gate (sum channel) surface potential is such that when S_T is turned on, the other charge Q still residing under S_1 cannot flow past the slave end of the floating gate and into gate S_s when S_s is turned on.

After this time interval is over, gate C_3 is turned on and the charge Q contained in the master end is then processed into C_τ and detected as an output, while the charge contained under S_1 and S_T, which was blocked by the slave load end of the floating gate is drained out by turning on gate GRD.



a) Output When Input is (1,1)



b) Output When Input is (1,0) or (0,1)

Figure 4-4. Two-Input Adder

4.2.1 Detail Analysis of Two-Input Adder

Determination of Constants

In the equations depicted in Section 2.4.4, the following constant are assumed and used throughout: $N_a = 10^{15}/cm^3$, $T = 300^oK$, $ni = 1.4 \times 10^{10}/cm^3$ <100> material, and $Q_{ss} = 1.4 \times 10^{-8}/cm^2$ (C_{oul}).

For metal - SiO_2 - Si (n-type) structure, flat band voltage

$$V_{FB} = \phi_{MS} - \frac{Q_{ss}}{C_{ox}}$$

becomes

$$V_{FB} = -0.3\ V - \frac{1.4 \times 10^{-8}/cm^3}{C_{ox}} \qquad (4.2)$$

For poly - SiO_2 - S_i (n-type) structures, flat band voltage

$$V_{FB} = +0.6 - \frac{1.4 \times 10^{-8}/cm^3}{C_{ox}}$$

$$V_o = \frac{e\ \varepsilon_s\ N_A}{C_{ox}^{\ 2}}$$

becomes

$$V_o = \frac{1.7 \times 10^{-16}}{C_{ox}^{\ 2}}\ \frac{C_{oul}}{cm^4} \qquad (4.2)$$

$$C_{ox} = \frac{0.2\ pf}{1000\mathring{A} - (mil)^2} = \frac{0.03 \times 10^{-6}\ f}{1000\mathring{A} - cm^2} \qquad (4.3)$$

Determination of Amount of Charge Q

The amount of charge Q can be calculated by using equation (2.3) of Section 2.4.4 which, when $q_o = 0$ (no fat zero), becomes

$$\Delta\phi_s = \frac{-\Delta Q}{C_{ox_2}} \left\{ 1 - \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_2}\right)}{V_o} \right]^{1/2}} \right\} \qquad (4.4)$$

From the V_G/ϕ_s curves (Figure 4-5) with a gate voltage of $V_G = -7$ volts being applied to C_2, and making the surface potential under gate C_1 equal to -2.5 volts by supplying -7 volts to its gate, the amount of charge becomes

$$Q = 0.264\ X\ 10^{12}\ Coul$$

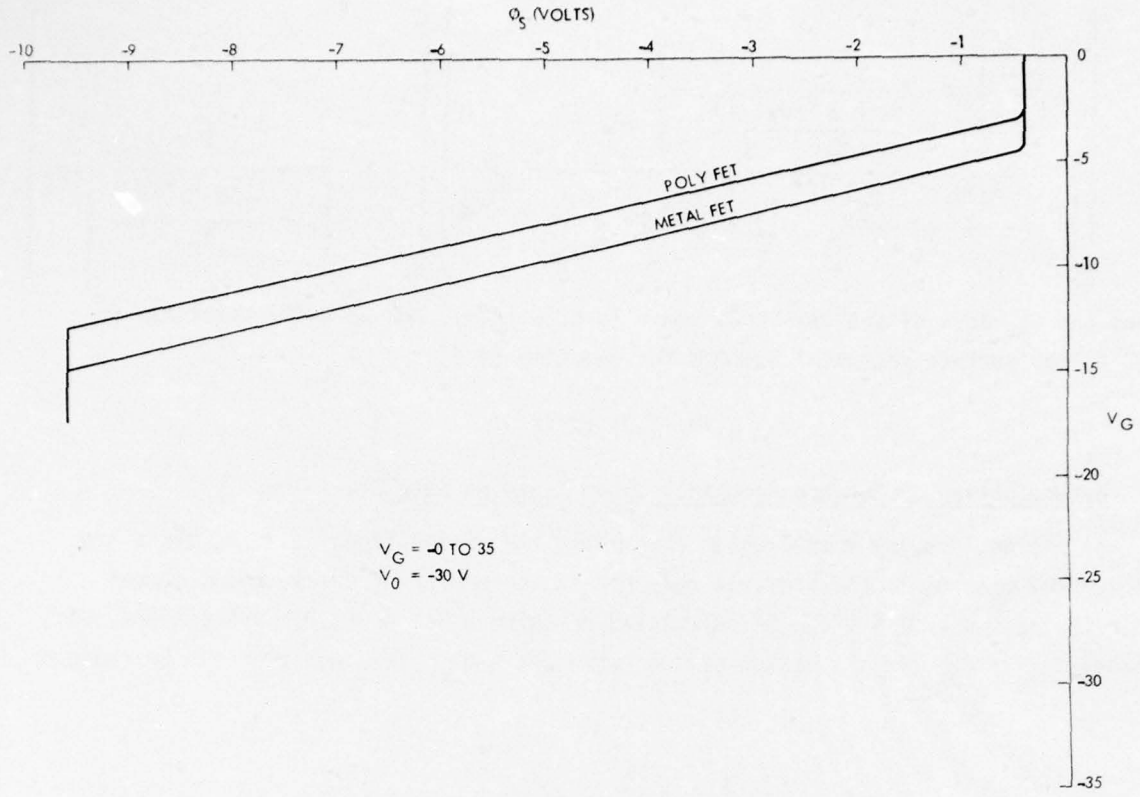


Figure 4-5. V_G/ϕ_s Curves for DP1-2 Wafer No. 5 (+10 V Bias)

This assumes for poly gate C_2 an oxide thickness of 1000 Å and an area of 0.78 $(mil)^2$.

Determination of Surface Potential Under Gate S_1

When the above charge Q is being dumped under gate S1, which is being subjected to a gate voltage of -9 volts, and a surface potential of -6 volts is determined from the V_G/ϕ_s curve, S_1 surface potential will change to a new value determined by

$$\Delta\phi_s = \frac{-\Delta Q}{C_{ox_2}} \left\{ 1 - \frac{1}{\left[1 + \frac{2\left(V_G - V_{FB_2}\right)}{V_o}\right]^{1/2}} \right\} \qquad (4.5)$$

which becomes after substitution

$$\phi_s = -4.49 \text{ volts}$$

Determination of Surface Potential Under Floating Gate Master End

As the charge Q is being dumped under the floating gate, after it has been subjected to a gate voltage of -17 volts which produces surface potential of -13.5 volts (as determined from the V_G/ϕ_s curves), its new surface potential is calculated by the following equation:

$$\Delta\phi_s = \left[\frac{\frac{\Delta Q}{C_{ox}} \left\{ \frac{1}{\left[1 + \frac{2(V_G - V_{FB_2})}{V_o}\right]^{1/2}} - 1 \right\}}{\left[1 + \frac{2(V_G - V_{FB_2})}{V_o}\right]^{1/2} \left(1 + \frac{C_{ox_4}}{C_{ox_2}}\right) + \frac{C_T}{C_{ox_2}}} - \frac{\Delta Q}{C_{ox_2}} \right] \left[1 - \frac{1}{\left[1 + \frac{2(V_G - V_{FB_2})}{V_o}\right]^{1/2}} \right] \quad (4.6)$$

Since the C_{ox} area of the two-input adder is 0.15 $(mil)^2$ and an oxide thickness of 1000 Å, the surface potential becomes for equation (4.6)

$$\phi_s = -5.36 \text{ volts}$$

Determination of Surface Potential Under Floating Gate Slave End

As charge is being dumped under the master end of the floating gate, the slave end surface potential will vary, in response to the amount of charge being dumped under the master end, and can be calculated by using equation (2.7) of Section 2.4.4. Because $C_{ox_2} = C_{ox_4}$ in the design of the two-input adder, this equation can be reduced to:

$$-\Delta\phi_s = \frac{\left(\frac{\Delta Q}{C_{ox_2}}\right) \left\{ \frac{1}{\left[1 + \frac{2(V_G - V_{FB_2})}{V_o}\right]^{1/2}} \right\}^2}{2 \left\{ \frac{1}{\left[1 + \frac{2(V_G - V_{FB_2})}{V_o}\right]^{1/2}} \right\} + \frac{C_T}{C_{ox_2}}}$$

and becomes

$$\phi_s = -11.84 \text{ volts}$$

Determination of Amount of Charge Contained Under S_1 and S_T

To ensure proper operation of the two-input adder, the amount of charge contained under gates C_1 and C_T simultaneously must be less than the amount of charge generated by C_2, providing that the surface potentials are contrained to the surface potentials of those determined by the change in the floating gate slave end. This calculation has to be done in two steps because S_1 gate is a poly gate with oxide thickness of 1000 Å, and S_T a metal gate with oxide thickness of 2000 Å. Taking these oxide

thicknesses and the surface potential into consideration; as depicted in Figure 4-3 under the floating gate, equation (4.8) below will determine the amount of charge Q, contained under gates S_1 and S_T.

$$\Delta\phi_s = \left(\frac{-\Delta Q}{C_{ox}}\right)\left\{1 - \frac{1}{\left[1 + \frac{2(V_G - V_{FB})}{V_o}\right]^{1/2}}\right\} \qquad (4.8)$$

This calculation yields a charge of

$$Q = 0.3668 \times 10^{-12} \text{ (Coul)}$$

which is greater than the charge generated by gate C_2 which was determined to be equal to 0.264×10^{-12} Coul.

In solving the above equations it is assumed that V_G is a constant of integration. This constant introduces approximately 3 percent error in the calculation, which is well within experimental error in determining the voltages from the V_G/ϕ_s experimental curves.

4.3 DETAIL DESCRIPTION OF THREE-INPUT ADDER CIRCUITS

The designs of the half-adder and full-adder logic cells described in Section 2.3.4 are very different from the full-adder incorporated on the DP-0 chip. It therefore seemed desirable to lay out a full-adder, a half-adder, and a two-input AND gate on the DP-1 chip. To assist in the analysis of test results it was decided to also place a floating gate amplifier on the chip. The full-adder and half-adder have separate test areas, but the floating gate amplifier and two-input AND gate are combined in a single test area. All three test areas have the same 22 bonding pad layout, and wherever possible the identical bonding pads are used for the same function on each of the three test cells. Therefore, it is possible to move directly from the AND gate to the half-adder and then to the full-adder without changing the test setup.

In the full adder logic cell layout shown in Figure 4-6; the polysilicon storage areas are marked to correspond to the schematic diagram of Figure 2-21.

During the preliminary layout of the multiplier and adder arrays it was found that in order to reach the inside logic cells with the large number of clock lines required, it was necessary to make many crossings of CCD channels located at the array perimeter. Since the only way of crossing a CCD channel with an aluminum line is over a polysilicon gate, the number of crossovers required governs the length of the polysilicon gates. In the preliminary layout the optimum layout required a maximum of three aluminum crossovers of some polysilicon gates. In the design rules used for the DP-1 layout, the aluminum width and spacing are both 0.3 mil. The three metal line

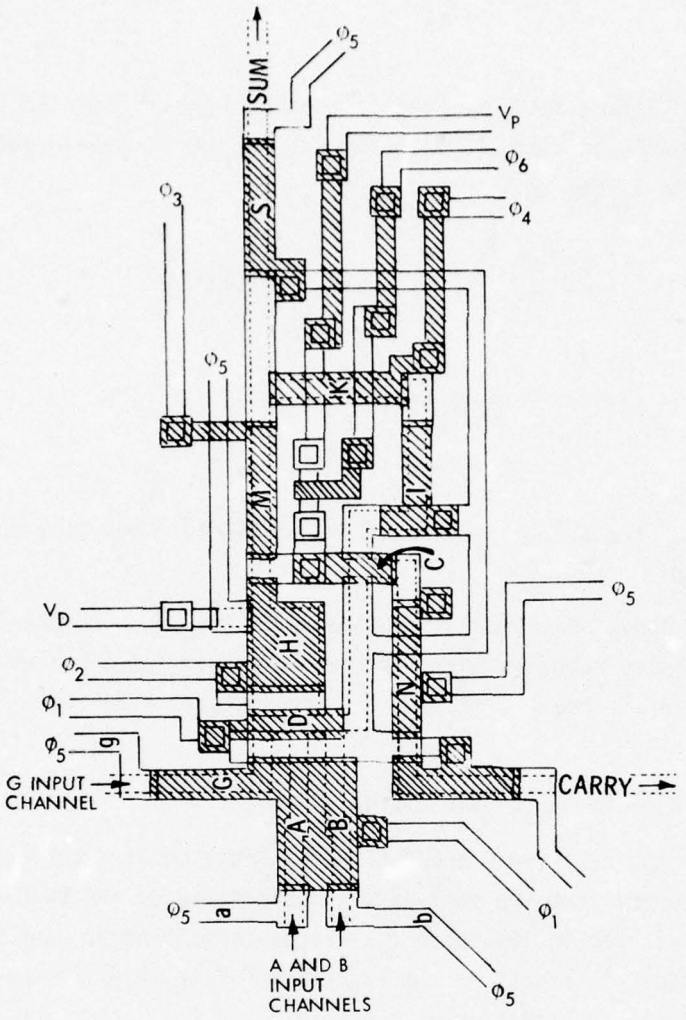


Figure 4-6. Composite Layout Diagram of the Full-Adder Logic Cell

widths separated between two metal gates require a polysilicon bridge of 7(0.3) = 2.1 mil. In addition, the aluminum gates overlap the polysilicon gate 0.1 mil at each end; so the polysilicon gate has to be increased to allow for the overlap; this makes the polysilicon gate length 2.3 mil.

Since the CCD channel width used on the DP-1 is 0.3 mil wide, the area of a polysilicon gate that is also used as a bridge for three aluminum crossovers is 0.69 mil^2.

In an attempt to keep as many storage well sizes and clock-line drive requirements as similar as possible, it was decided to make the 2.3 x 0.3 polysilicon gate area the standard throughout the layout of the three test cells and arrays.

In the full adder layout shown in Figure 4-6 the polysilicon areas labeled A, B, G, D, M, I, K, S, N all measure 2.3 x 0.3 mil. The polysilicon area C under the floating gate was made smaller in order to obtain as large a charge in V_s as possible and measured 1.0 x 0.3 mil.

It was shown in Section 2.3.4 that in order to provide adequate noise margins when the standard charge stored in D cell was transferred to the H storage cell, it was necessary to spread out the charge by making the H area 1.8 mil^2. From Figure 4-6, the H storage area is in fact approximately twice the standard storage area.

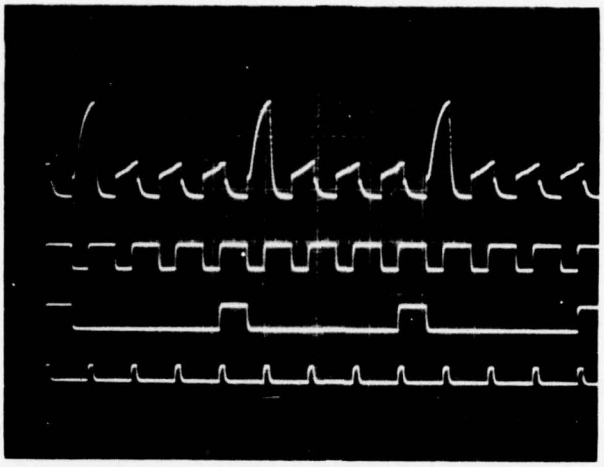
In addition to the need to make crossovers within a logic cell, the layout of the multiplier and adder arrays also requires that crossovers be made in between logic cells. This is made possible by placing one shift-register stage-delay in the carry-bit line between two full-adders. The polysilicon gate of the single stage shift-register allowed three crossovers which was found to be adequate.

The single-bit stage delays can be seen in the block diagrams of Figures 2-26 and 2-27. Due to insertion of delay stages to provide crossovers, it was also necessary to add extra delays in the sum output lines to ensure that the output bits remained synchronous.

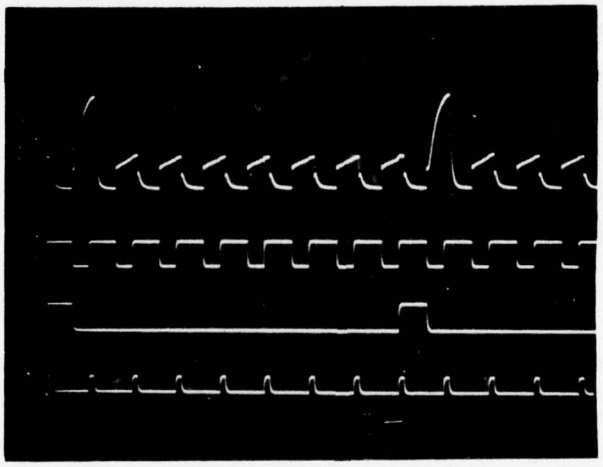
The two-word 4-bit adder array and the two-word 3-bit multiplier array were laid out within the same 33 pad configuration; this enabled testing to be performed with a single 33 probe card. The layout of the three test cells and two arrays is shown in Figure 4-1. In order to compensate for transfer losses within sequential individual three-input adders of the adder array, the Ø1 clock line to each three-input adder was taken to a separate bonding pad. However, to carry out the same procedure on the multiplier would have made the clock line interconnect pattern too complex for an evaluation array.

It was decided to test the different test devices and arrays in ascending complexity, thus the two-input AND gate was the first device tested. The schematic diagram of the AND gate (Figure 2-24) is a very simple logic cell, requiring only a two-phase clock. No difficulty was experienced in verifying correct operation of this cell, and output signals of approximately one volt amplitude were produced (Figure 4-7).

The half-adder test cell involves the use of a floating gate and is therefore more complex than the AND gate. The schematic diagram of the half-adder is shown in Figure 2-23. Correct operation of the half-adder was verified with clock phases as shown in Figure 2-22 and at clock amplitudes within 10 percent of those predicted in Section 2.4.1. A photograph of the input and output waveforms of the half-adder logic cell is shown in Figure 4-8.



AND Output Signal	1 V/cm
"a" Input Signal at f	5 V/cm
"b" Input Signal at f/4	5 V/cm
Inject Diode Waveform	5 V/cm



AND Output Signal, Scale	1 V/cm
"a" Input Signal at f	5 V/cm
"b" Input Signal at f/8	5 V/cm
Inject Diode Waveform	5 V/cm

Figure 4-7. Input and output signals of the two-input AND gate showing that an output bit is generated only when both input bits are at logic "1" (positive pulses = "1") and a correct half cycle delay

Sum Output Signal, Scale 1 v/cm

Carry Output Signal, Scale 1 v/cm

"A" Input Signal

"B" Input Signal

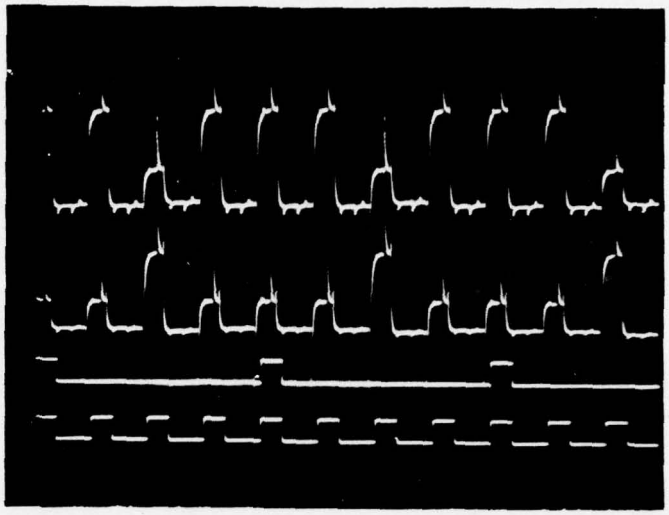
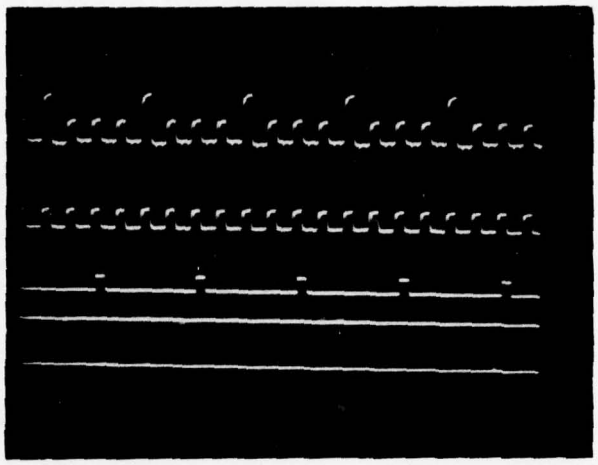


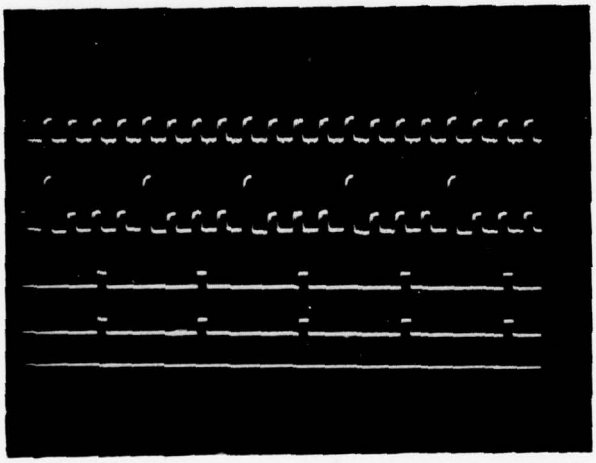
Figure 4-8. Input and Output Signals of Half-Adder Gate, Showing a Sum Output Each Time A or B Equal "1", But Not When Both Equal "1"

The full-adder is the most complex of the individual test devices, having three inputs and three outputs. Two of the output channel paths converge to form an OR function as shown in Figures 2-21 and 4-6.

The full-adder was shown to function with the same clock phases and clock amplitudes as the half-adder. Photographs of the input and output waveforms of the full-adder are shown in Figure 4-9. In Figure 2-22 the phase relationship of Ø4 and Ø5 are shown to be identical, the two waveforms only differ by amplitude; Ø4 switches from -4.5 to -8.7, whereas the Ø5 clock line switches from -4.5 to -24. As an experiment the Ø4 clock line was removed and the K gate connected to Ø5; the full-adder continued to function correctly and the amplitude of the output signals was unchanged.

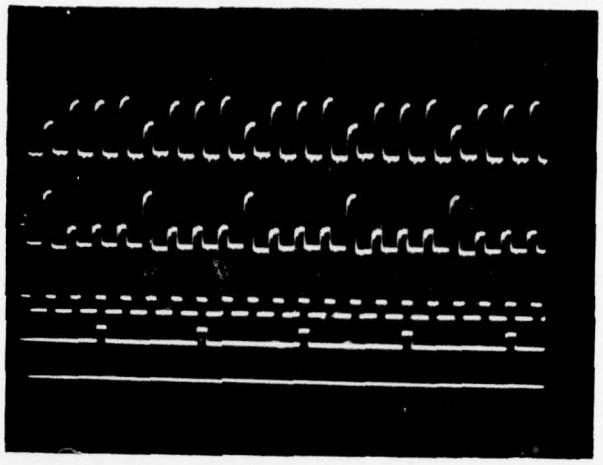


Sum-Bit Output
Carry-Bit Output
A Signal Input
B Signal Input
G Signal Input

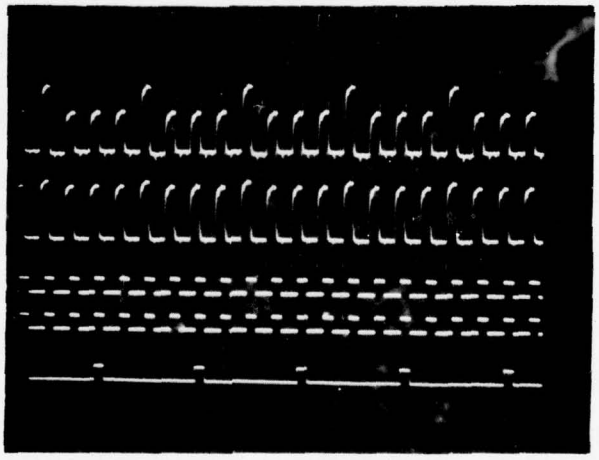


Sum-Bit Output
Carry-Bit Output
A Signal Input
B Signal Input
G Signal Input

Figure 4-9. Input and Output Waveforms Obtained from Full-Adder Test Cell



Sum-Bit Output
Carry-Bit Output
A Signal Input
B Signal Input
G Signal Input



Sum-Bit Output
Carry-Bit Output
A Signal Input
B Signal Input
G Signal Input

Figure 4-9. Input and Output Waveforms Obtained from the Full-Adder Test Cell (Continued)

4.4 ARRAY FUNCTIONAL TESTING

4.4.1 Functional Testing of the 4-Bit Adder-Array

During the testing of the multiply and add arrays that use the full-adder cell, several mask errors were found. However, it was possible to circumvent these errors and sufficient testing was carried out to show that both arrays performed their arithmetic functions correctly.

Two mask-errors were discovered on the 2-word, 4-bit adder array; one of the errors was the omission of two windows in the TEOS etch mask. The omission resulted in two metal gates having a layer of TEOS oxide between the aluminum gate and the gate oxide, preventing the gates from controlling the channels. This error was circumvented by not carrying out the TEOS process step and risking the high probability of metal breakage.

The second mask error found on the adder array was the omission of the aluminum conductor from a contact hole associated with the least significant output bit, resulting in a logical "0" output under all input combinations. This error was circumvented by only using input words in which both the least significant bits were the same, thus their sum is always a logical "0". Note that the proper operation of the logic circuits in the least significant channel can be verified with this approach.

The adder-array is designed to add any two 4-bit binary numbers, from 0000 to 1111 (decimal 30), together and produce a 5-bit binary sum of value from 00000 to 11110 (decimal 60).

Testing was initially carried out at room temperature (25°C) and at a clock frequency of 10 kHz. The clock frequency was divided down by 16 to produce a 625 Hz word rate so that only one output word was displayed on the monitoring CRT at one time. By using this technique we could check that the phase correcting shift register stages had been inserted correctly since all output bits should be coincidental in time.

The six photographs of Figures 4-10 through 4-15 show the 4 most significant bits of the output sum for various input number combinations.

4.4.2 Maximum Clock-Rate of the 4-Bit Adder Array

The maximum theoretical clock rate of the adder-array is determined by the thermal diffusion time along the longest length of electrode along which a charge is moved from one potential well to the next. In the full-adder design, the $\phi 1$ electrode that controls the three storage areas D, C, and I has the maximum length. Taking into consideration that the original layout was multiplied by 1.5, the length of the $\phi 1$ gate is 10.08 mil.

The thermal diffusion time can be determined from;

$$\tau = \frac{4 L^2 q}{\pi^2 k T_\mu}$$

[Not shown, S_1 = 0]

Output bit S_2 = 1

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 0

(Decimal 28)

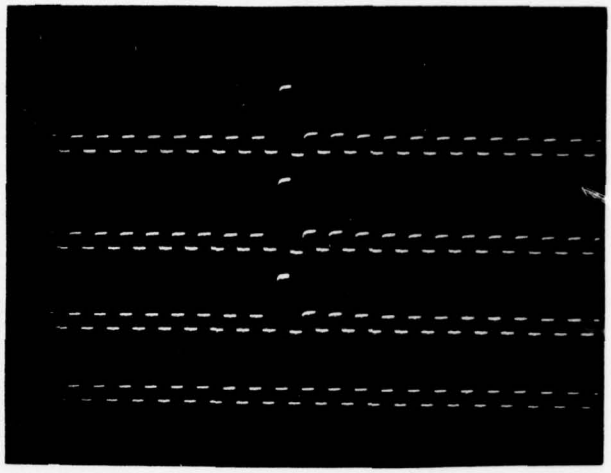


Figure 4-10. Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0000 (0)

[Not shown, S_1 = 0]

Output bit S_2 = 0

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

(Decimal 32)

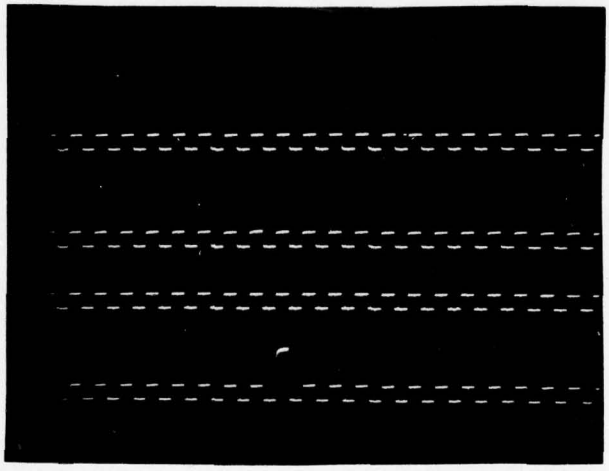


Figure 4-11. Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0010 (4)

[Not shown , S_1 = 0]

Output bit S_2 = 1

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

(Decimal 36)

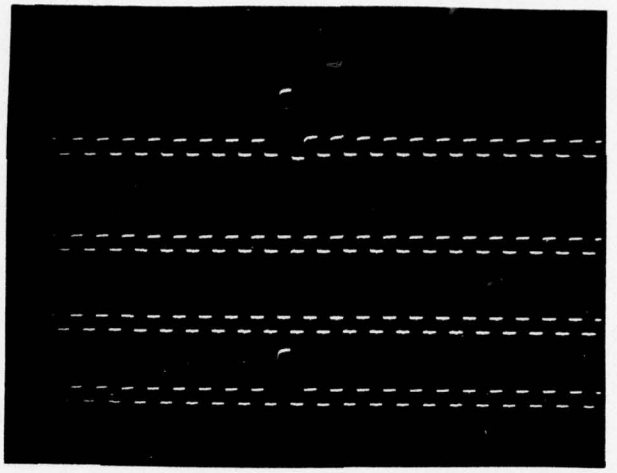


Figure 4-12. Output Signals from the 2-Word, 8-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 0100 (8)

[Not shown, S_1 = 0]

Output bit S_2 = 1

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 1

(Decimal 60)

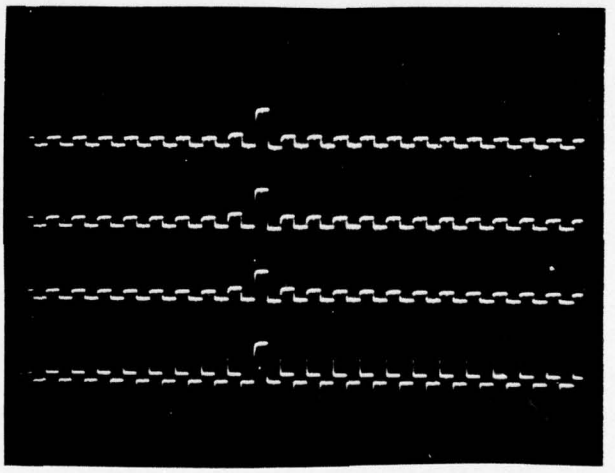


Figure 4-13. Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1111 (30) and the b-Word = 1111 (30)

[Not shown, S_1 = 0]

Output bit, S_2 = 1

Output bit, S_3 = 0

Output bit, S_4 = 1

Output bit, S_5 + 0

(Decimal 20)

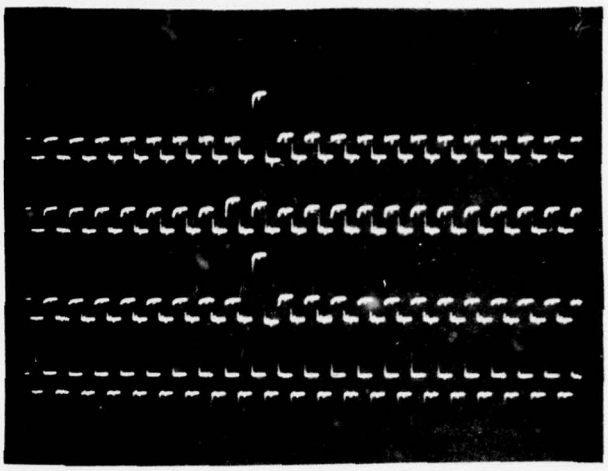


Figure 4-14. Output Signals from the 2-Word, 4-Bit adder Array When the a-Word = 0101 (10) and the b-Word = 0101 (10)

[Not shown, S_1 = 0]

Output bit, S_2 = 1

Output bit, S_3 = 1

Output bit, S_4 = 0

Output bit, S_5 = 1

(Decimal 44)

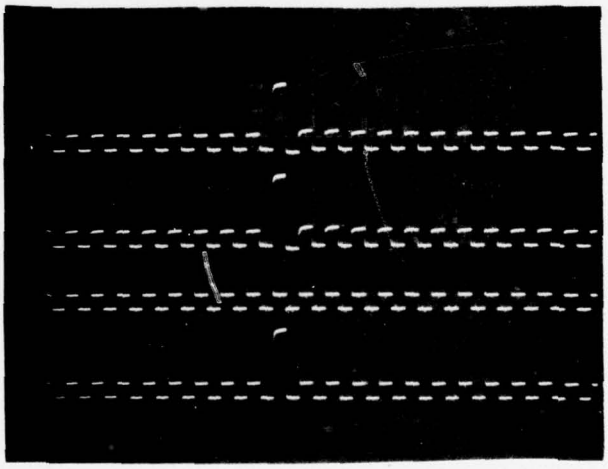


Figure 4-15. Output Signals from the 2-Word, 4-Bit Adder Array When the a-Word = 1110 (28) and the b-Word = 1000 (16)

where the length of the transmission electrode is

$$L = 27.7 \times 10^{-3} \text{ cm}$$

$$\pi^2 = 9.87$$

$$\frac{kT}{q} = 0.0259 \text{ volts}$$

the mobility for $N_A = 10^{15}$ is

$$\mu = 5 \times 10^2 \text{ cm}^2/\text{V sec}$$

The waveform of the $\phi 1$ clock has a symmetrical aspect ratio, so that the thermal diffusion time is one-half the clock period and so the maximum theoretical clock rate is

$$f_{max} = \frac{1}{2\tau}$$

$$= \frac{\pi^2 \, kT \, \mu}{8 \, L^2 \, q}$$

$$= \frac{9.87 \times 0.0259 \times 5 \times 10^2}{8 \times 7.67 \times 10^{-4}}$$

$$= 20.8 \text{ kHz}$$

Testing on a probe station at clock rates above 11 kHz proved very difficult due to the clock pulse coupling between the probes. Wafer DPI, lot 3 No. 9 was initially tested on the probe station, mapped, diced, and several samples bonded. All of the following tests were performed with bonded chips. The outputs of the 3 most significant bits were photographed at an operating frequency of 11 kHz. Figure 4-16 shows the correct output 011 x x when the a-word = 1100 and the b-word = 0000. The b-word was then switched to 0100 and the 3 output bits correctly switched to 100xx as shown in Figure 4-17.

The clock frequency was then increased to 50 kHz and the b-word input switched as before; the array performed correctly as shown in Figures 4-18 and 4-19. Note the outputs at 50 kHz are less than at 11 kHz. This follows the predicted limit of 20.8 kHz for the complete transfer of thermal carriers.

Output patterns for both of the input conditions were photographed at clock frequencies of 100 kHz and 175 kHz and the results are shown in Figures 4-20 through 4-23. Note that the logic "1" output levels do not attenuate significantly as the frequency is increased, however; they become obscured as the logic "0" output levels (fat-zero) grow larger.

Horizontal Scale IV/Div

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 0

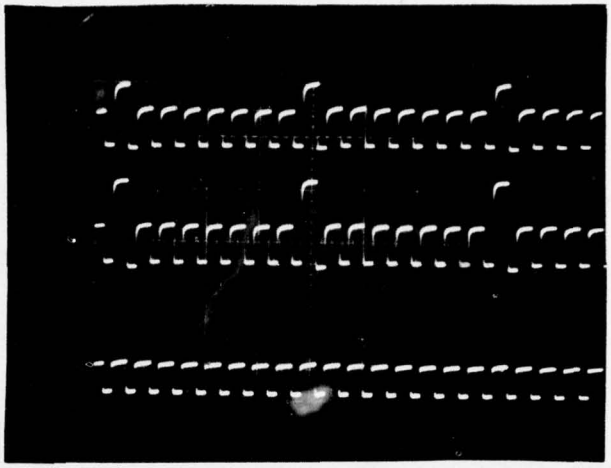


Figure 4-16. Three Most Significant Output Bits from the Adder Array Operating with a 11 kHz Clock and Inputs of a = 1100 and b = 0000

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

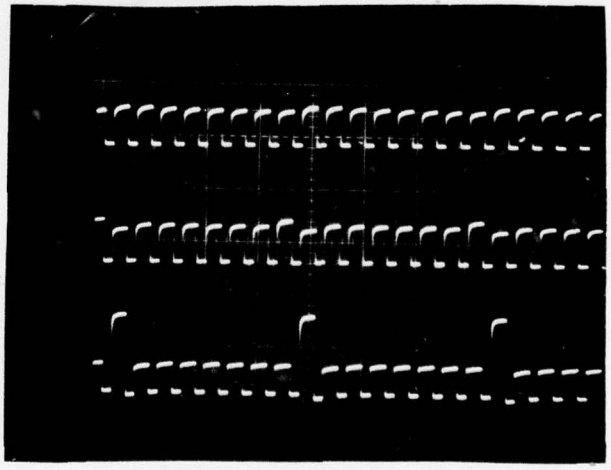


Figure 4-17. Three Most Significant Output Bits from the Adder Array Operating with a 11 kHz Clock and Inputs of a = 1100 and b = 0100

Horizontal Scale IV/Div

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 0

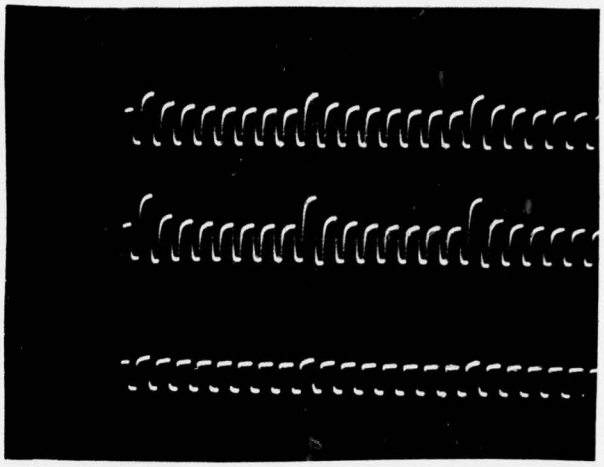


Figure 4-18. Three Most Significant Output Bits from the Adder Array Operating with a 50 kHz Clock and Inputs of a = 1100 and b = 0000

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

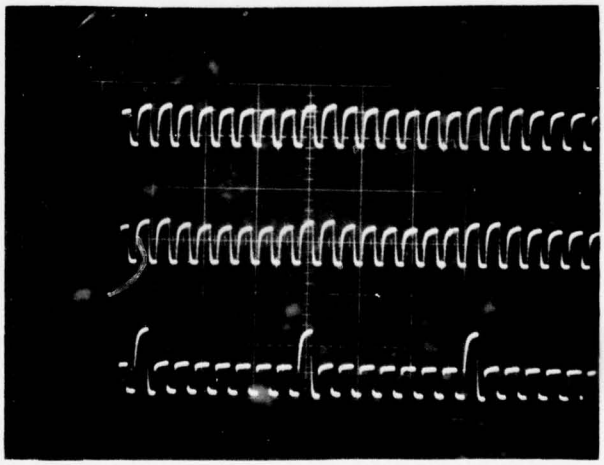


Figure 4-19. Three Most Significant Output Bits from the Adder Array Operating with a 50 kHz Clock and Inputs of a = 1100 and b = 0100

Horizontal Scale IV/Div

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 0

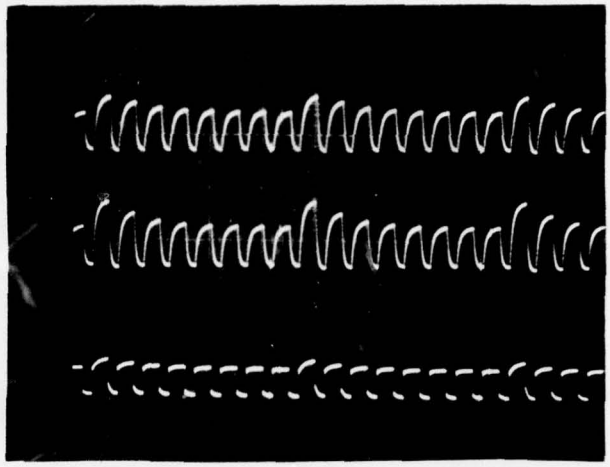


Figure 4-20. Three Most Significant Output Bits from the Adder Array Operating with a 100 kHz Clock and Inputs of a = 1100 and b = 0000

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

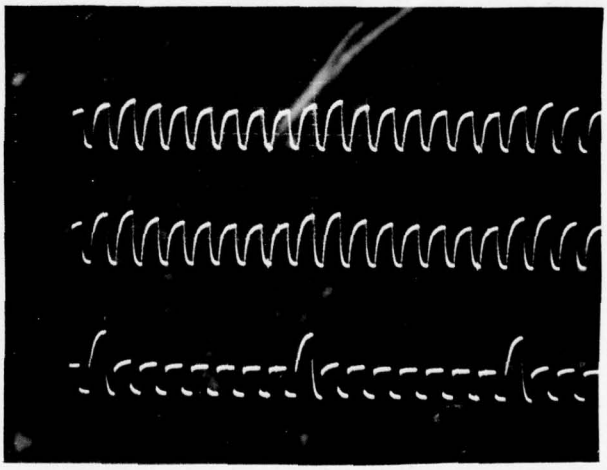


Figure 4-21. Three Most Significant Output Bits from the Adder Array Operating with a 100 kHz Clock and Inputs of a = 1100 and b = 0100

Horizontal Scale IV/Div

Output bit S_3 = 1

Output bit S_4 = 1

Output bit S_5 = 0

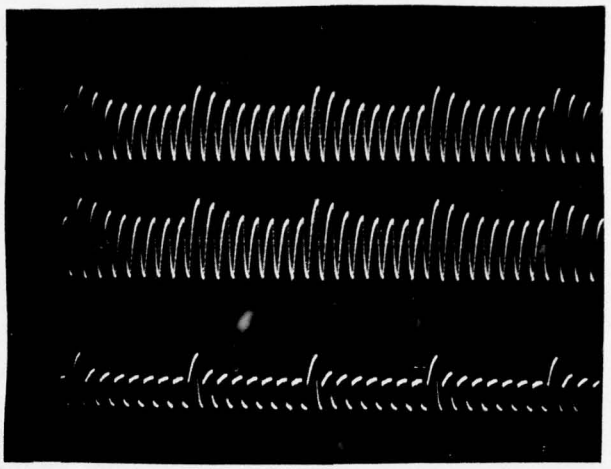


Figure 4-22. Three Most Significant Output Bits from the Adder Array Operating with a 175 kHz Clock and Inputs of a = 1100 and b = 0000

Horizontal Scale IV/Div

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

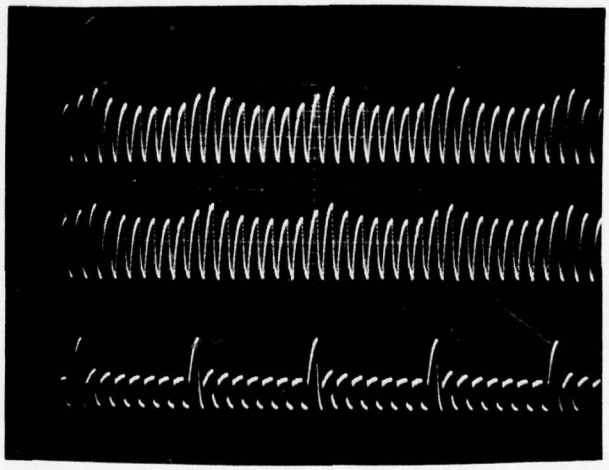


Figure 4-23. Three Most Significant Output Bits from the Adder Array Operating with a 175 kHz Clock and Inputs of a = 1100 and b = 0100

By switching the b_3 input and observing the output patterns on the oscilloscope, it can be seen that the adder array is performing the correct arithmetic functions up to just beyond 200 kHz, but with a deteriorated signal-to-noise level.

4.4.3 Operating Temperature Range of the 4-Bit Adder Array

The operating temperature range was determined by functional testing in a temperature controlled chamber.

The clock voltages were adjusted at a frequency of 11 kHz and at 25°C so that the 3 most significant output bits from the adder array were performing correctly for each input combination and with a maximum signal-to-noise ratio. The temperature was increased in 10° increments while the inputs were switched and the outputs monitored; no change in performance was observed at 35°C or 45°C.

At 55°C the fat-zero level of sum-bit S_4 and carry-bit S_5 increased; sum-bit S_3 output remained unchanged.

At 65°C the full-adder in the fourth channel ceased to switch, the S_4 sum-bit output remained at "0", and the S_5 carry-bit output remained at "1". The S_3 sum-bit output continued to function correctly.

At 75°C the S_4 sum-bit output inverted and the fat-zero level of S_3 began to increase.

At 85°C no further change. At this point the voltage on the control gate C_2 was adjusted in an attempt to reduce the fat-zero, but this was unsuccessful.

At 95°C no further change.

At 105°C the S_3 sum-bit output inverted.

At 110°C the fat-zero level of all outputs had increased so that no signals were discernable. It should be noted that the combination of high temperature and low frequency is the most difficult operating condition from the standpoint of thermal leakage. Indeed, proper operation at 125°C could be assured simply by operating the existing device at a frequency above about 500 kHz.

The temperature was then reduced to 25°C and all outputs resumed operating correctly. The temperature was then lowered in 10° steps, the inputs switched and again the 3 most significant output bits monitored. No change in performance was noted at 15°C or 5°C.

At -5°C, the S_4 and S_5 outputs ceased to switch correctly with some input combinations, indicating that the carry-bit from the second-most-significant full-adder was not being transferred.

At -15°C to -55°C the fat-zero level was reduced, but no further change in arithmetic performance. The temperature was then reduced to -65°C and the C_2 control line adjusted

so that all channels performed correctly with a maximum signal-to-noise ratio. Figures 4-24 and 4-25 show the 4 most significant output bits from the adder-array for two different input combinations when the chip is operating at -65°C.

[Not shown, S_1 = 0]

Output bit S_2 = 1

Output bit S_3 = 0

Output bit S_4 = 0

Output bit S_5 = 1

(Decimal 36)

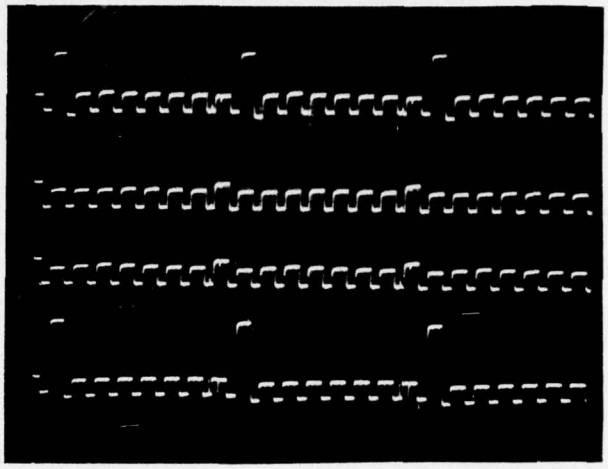


Figure 4-24. Output Signals from the 2-Word, 4-Bit Adder Array when a = 0100 (8) and b = 1110 (28) Operated at -65°C

[Not shown, S_1 = 0]

Output bit S_2 = 0

Output bit S_3 = 1

Output bit S_4 = 0

Output bit S_5 = 1

(Decimal 40)

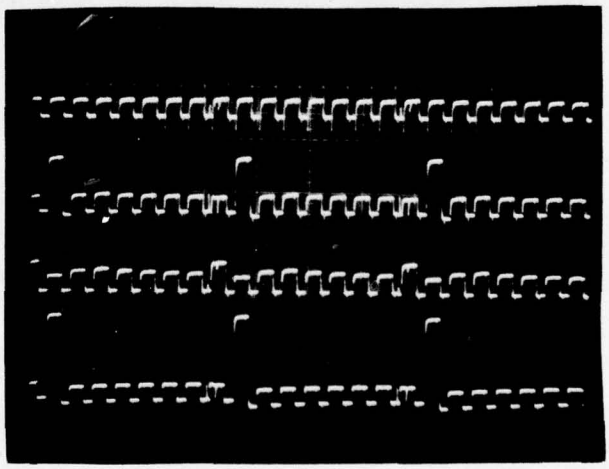


Figure 4-25. Output Signals from the 2-Word, 4-Bit Adder Array when a = 1010 (20) and b = 1010 (20) Operated at -65°C

4.4.4 Functional Testing of the 3-Bit Multiplier Array

Following the demonstration of the adder array, the 2-word, 3-bit multiplier array was also functionally tested. Two mask errors were found which limited its operation. One was a 0.05 mil discontinuity in a metal conductor carrying an input signal and the other error was the omission of a connection to a polysilicon gate in the sum channel of the most significant full-adder. However, since the carry output of the full-adder was correctly connected and functional, it was possible to demonstrate all channels in the multiplier.

By keeping the a_1 input to the open conductor at logic "0" and exercising all the other five inputs through all possible combinations, it was possible to completely demonstrate the correct logic operation of the entire multiplier array.

There are six parallel outputs from the multiplier, and 196 different output numbers, so that producing a meaningful photograph showing simultaneous outputs is quite difficult. For simple combinations where only four outputs are changing, the output pulses are similar to those shown for the adder-array. However, when all logic cells are operating and charges have to propogate through several full-adder, there is some deterioration of the charge and a difference in the amplitude of the most significant output bits can be observed. The decrease in amplitude being dependent on the number of sequential charge buckets. Nevertheless, correct operation for all input combinations was demonstrated.

4.4.5 Interfacing the CCD Devices to TTL Drivers

It is very easy to interface CCD data processing devices to TTL logic gates; this was demonstrated by adding +10 volts to the substrate bias and all of the clock and control lines, as shown in Figure 4-26.

The standard output swing of ground to 4.5 volts from the TTL gate is connected directly to an a, b, or g input gate on the full-adder or other CCD device. The phase relationship and amplitude of the various clock waveforms is shown in Figure 4-27; they are referred to the complete schematic of the full adder shown in Figure 4-24.

Note that amplitude and tolerances for the clock pulses and bias lines are shown in Figure 4-27 for a particular device.

4.4.6 Characterization Summary

The data derived from the tested devices is summarized in Table 4-2. The circuit designs tested here were produced to demonstrate the functionality of the arrays and consequently no effort was made to optimize the design. The relatively low operating frequency is a direct result of the maximum gate length used. The maximum length is much longer than required and future designs will be built for operation in the low megahertz range. It has already been mentioned that the relatively low test frequency

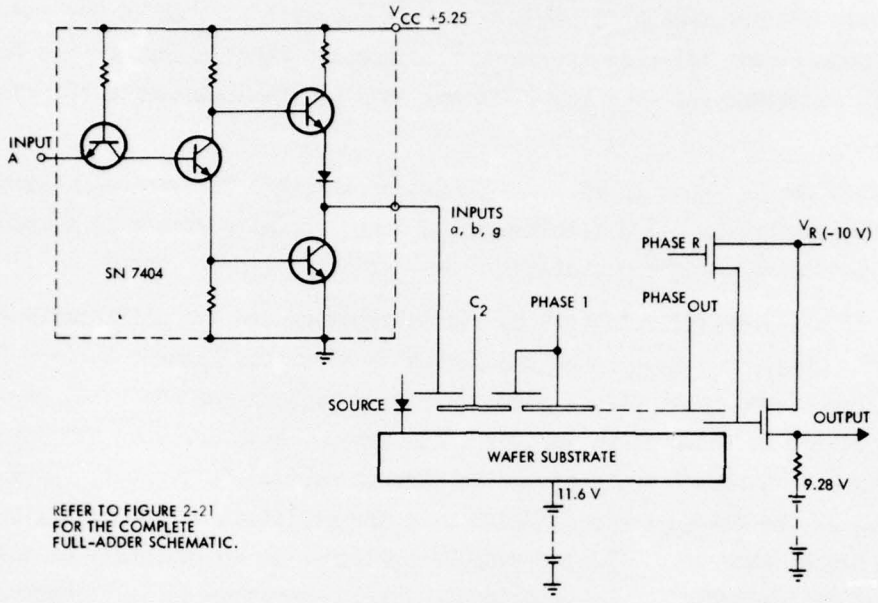


Figure 4-26. Interconnections and DC Voltage Levels Required to Drive a CCD Signal Processing Device from a TTL Gate

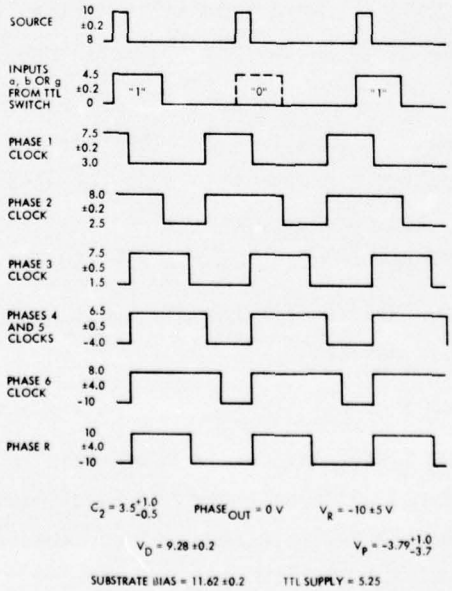


Figure 4-27. Amplitudes and Phase Relationship of Clock and Data Waveforms for TTL to Adder Array Compatibility

Table 4-2. Preliminary Characteristics of Existing Design

Frequency Range
Design goal: 20 kHz
Maximum frequency with no output degradation: 50 kHz
Maximum frequency with correct compution: 200 kHz
Maximum Temperature
At 11 kHz: +65°C
At 700 kHz: +125°C*
Input/Output
Demonstrated to be TTL compatible
*Calculated from 11 kHz data.

(11 kHz) would be expected to impose a low maximum temperature. The low frequency allows more time for thermal carriers to accumulate and eventually cause an error in the output. Normally, the devices are expected to operate near 1 MHz and at this frequency even the existing design would operate correctly at temperatures exceeding +165°C. In future designs the total device area will be reduced and so the maximum operating temperature at 1 MHz would be even higher; viewed another way, at any given temperature the reduced area devices will allow operation at lower frequencies.

5. PROCESS

5.1 PROCESS DESCRIPTION

The substrates are N-type silicon wafers with a resistivity of 3 to 5 Ω/cm and <100> orientation to minimize the surface charge. Initially, a 15,000 Å oxide is grown by a dry oxygen, wet nitrogen, dry nitrogen oxidation cycle. The pattern for the channel is photoresisted, the oxide etched, and a 1000 Å gate oxide is grown at 920°C in a H_2O atmosphere. This step is followed by the deposition of a 3500 Å polysilicon film which is phosphorous doped by gaseous diffusion. Next the polysilicon is covered by a Si_3N_4 film which is slightly oxidized. This step is followed by photoresisting and defining the polysilicon film by plasma etching. Thereafter an oxidation is performed for thickening the oxide existing in the channel region. This operation is required to ensure sufficient masking oxide in the channel for preventing boron penetration during the source and drain diffusion which is carried out after the source and drain pattern is photoresisted and the oxide etched. Next the Si_3N_4 covering the polysilicon film is etched followed by etching the channel oxide. After this step, a 2000 Å thermal oxide film is grown covering both the channel region (under the Al) and the polysilicon layer, contacts are defined, the oxide is etched, Al is deposited, and the interconnections are defined by a photoresist step. Finally the circuits are subjected to a 450°C sinter in N_2 followed by a 450° sinter in H_2. It should be noted that positive photoresist is utilized through all the processing.

This process is the basic Si-gate process which has been utilized with the DP-0 mask set. A metal step coverage problem was identified in this process. It is caused by the formation of a "gulch" under the polysilicon conductors in the field oxide region during the etching of the 1500 Å oxide covering the "metal" channel region (Figure 5-1) and by the steepness of the polysilicon film edge. This kind of step is very difficult to cover by the Al metallization producing open metal lines. It was

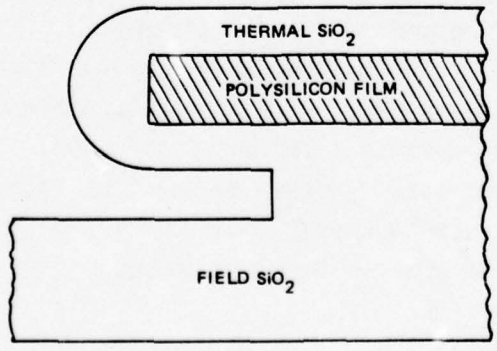


Figure 5-1. "Gulch" Formed Under the Polysilicon Film

found at TRW that films formed by thermal decomposition of tetraethyl ortho silicate (TEOS) produce very smooth covers in steps of the kind just described (see Figure 5-2). To utilize this TEOS deposition, an extra mask level had to be incorporated in the process sequence. Thus the DP-1 set of masks had this extra level which permits the oxide etching in the channel region while at the same time protecting the field oxide region which is where the step coverage problem occurs.

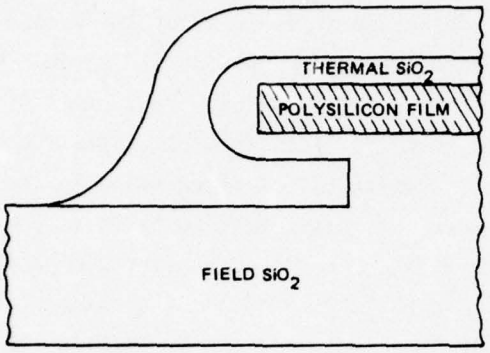


Figure 5-2. "Gulch" and Steep Polysilicon Step Covered by a TEOS Film

5.2 SPECIAL PROCESSES

5.2.1 Clean Gate Oxide Technology

TRW has developed clean oxidation methods to form SiO_2 films on silicon in the fabrication of the gate structure of CCD devices. It has been well established that three major processes affect the electrical stability of thermally grown oxides: substrate cleaning, oxidation, and metallization. Elaborate cleaning and metallizing techniques have therefore been developed. However, the oxidation itself is the most critical step by far. Early experiments using conventional oxidation systems showed that reproducible fixed charge values could be obtained only with new quartz tubes; however, these values deteriorated rapidly with time. Reproducible fixed charge values were found when double wall quartz tubes were utilized and, as shown in Figure 5-3, these values changed as a function of time of operation of the particular quartz tube, being more pronounced for <111> than for <100> oriented substrates. Faster aging was observed in oxides grown in spectrosil than in GE 204 quartz. It has been claimed that spectrosil quartz is practically free from metallic impurities (total concentration = 1 ppm); however, a recent analysis shows Na (20 ppm) as its main impurity. GE 204 contains Al (40 to 50 ppm) and Na (20 to 30 ppm).

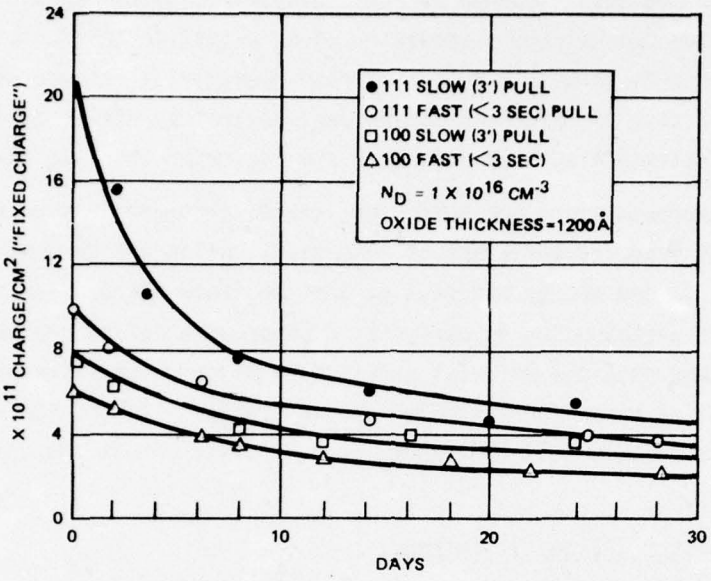


Figure 5-3. Fixed Charge as a Function of Quartz Tube Age

TRW's results also indicate that by very fast quenching to room temperature, the fixed charge can be eliminated. Moreover, when oxides with small values of the fixed charge (1×10^{11} cm^{-2}) were quenched in N_2 to -20°C, or annealed in N_2 at a low temperature (550°C), the fixed charge could be eliminated and experimental curves near the theoretical values obtained. It is suspected that Al and Na atoms in the oxide originating in the quartz tube are causing the observed aging phenomena. The quenching dependent changes of the fixed charge may be thus caused by an interstitial $\rightleftarrows$ substitutional reaction of Al atoms in the SiO_2 network lattice.

5.2.2 Ion Implantation

The impurity ion implant into semiconductors from a high energy accelerator (30 to 200 keV) has many attractive features for improved CCD manufacturing control. It offers a means of introducing precise measured quantities of a single species of doping impurity in a way that far surpasses what can be done with a thermal diffusion furnace. The accelerator is used as an adjunct to the conventional methods and the process still involves impurity distribution by thermal means. TRW has in operation a 200 keV ion accelerator which is presently set up to implant boron, phosphorus, and arsenic.

The ion dose may be controlled as accurately as the following factors will allow. These are the degree to which the ions are single charged, effectiveness of suppression of secondary electrons, and electronic limitations of the integration system.

Uniformity of the dose produced by ion implantation is determined by linearity of the beam scanning process and suppression of unscanned neutrals. Linearity of the scanning process is largely a matter of good electronics design, while suppression of neutrals is a matter of filtering out as many neutral particles as possible prior to deflection and maintaining good vacuum in the region of the scan plates.

To a first approximation, implanted ions are distributed in a Gaussian shape. The mean value and standard deviation of this distribution are determined in a predictable manner from ion energy and mass of the substrate atoms. The major deviation from this Gaussian distribution is caused by a phenomenon called channeling, in which ions are guided deep into the material parallel to major crystalline axes. Channeling can be controlled by proper alignment of the crystal with the ion beam direction, or by implanting through a thin amorphous layer of oxide or nitride on the wafer surface.

5.2.3 Polycrystalline Silicon Technology

Polycrystalline silicon films are used as gate electrodes in our CCD technology. Owing to the self-aligning features of the processing and since they form a very critical portion of the device structure, the chemical and physical structure of these films has to be very well controlled. Thus deposition, doping, and etching have to be carefully determined. These films are usually deposited by the thermal decomposition of silane in rf-heated, horizontal epitaxial reactors in a hydrogen carrier gas. Their crystalline structure is very sensitive to deposition conditions. And it is the film's crystalline structure which determines its electrical characteristics. We have found at TRW that 650^{o}C is the optimum deposition temperature for polycrystalline films. Films prepared at higher temperature usually show poor crystalline perfection. The deposition rate at 650^{o}C is approximately 0.1 μ/minute. Since these films are used both as conductors and diffusion masks, no doping impurities are added intentionally. Conventional gaseous diffusion techniques have been applied to dope these films. Thus BBr_3 is used to diffuse boron and $POCl_3$ is used to diffuse P. The diffusion temperature is 950^{o}C.

5.2.4 TEOS

Silicon oxide films produced by the pyrolytic decomposition of tetraethyl ortho silicate have been used to produce masking oxides and thickening field oxides. The thermal decomposition is carried out at 725^{o}C in a low vacuum (1μ). Immediately after deposition, these films have relatively low density and their etch rate in conventional buffered hf solution is very high (10:1 compared to thermal oxides). They are usually "densified" at elevated temperatures to produce films comparable in etch rate to that of thermal oxides.

These films tend to produce very good step coverage over polysilicon steps, therefore preventing "shadowing" during deposition of the Al films used for pads and interconnections. Figure 5-4b shows a scanning electron microscope picture of polysilicon gates covered with TEOS as compared to the same kind of gate covered by silox deposition in Figure 5-4a. Clearly shown is the gentle slope obtained with TEOS as compared to the rounded oxides obtained with silox.

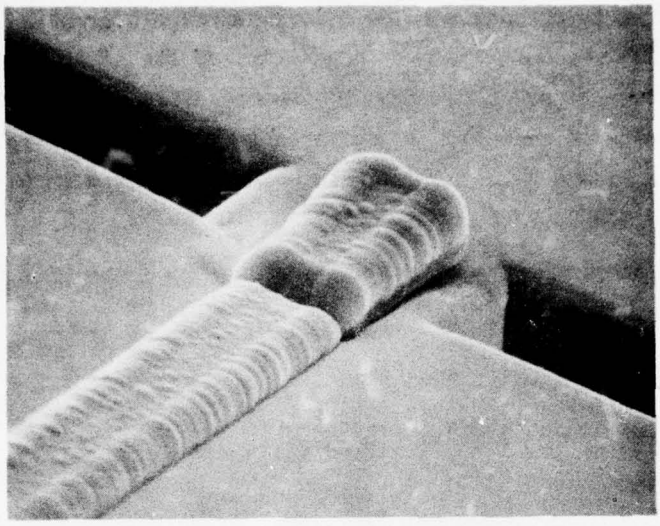
5.3 BASIC PROCESSING STEPS

The following is a list of the major process sequence steps:

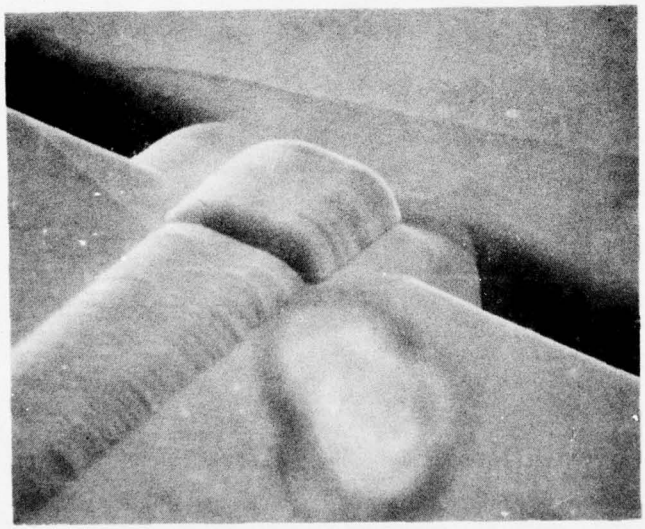
1. N-type wafers <100>, 3 to 5 Ω cm
2. Field oxide grown 15,000 Å
3. Photoresist channel
4. Etch channel oxide
5. Grow 1000 Å gate oxide (Figure 5-5a)
6. Deposit 3500 Å polysilicon film at 650°C
7. Diffuse P into the polysilicon; $PoCl_3$ source at 950°C
8. Deposit 2000 Å nitride
9. Photoresist polysilicon (Figure 5-5b)
10. Etch oxide
11. Plasma etch nitride and polysilicon films
12. Reoxidize channel oxide to 1500 Å
13. Photoresist source and drain
14. Etch oxide
15. Predeposit and diffuse boron; BBr_3 source at 950°C (Figure 5-5c)
16. Etch Si_3N_4.

Steps 1 through 16 are common to DP-0 and DP-1 processing. DP-0 processing continues as follows:

17. Etch oxide
18. Grow 2000 Å channel oxide (Figure 5-5d)
19. Photoresist contacts
20. Etch oxide
21. Evaporate Al
22. Photoresist and define Al (Figure 5-5e)
23. Sinter 450°C N_2 and H_2.



a) Covered by 13,000 Å Silox Film



b) Covered by 13,000 Å TEOS Film

Figure 5-4. Polysilicon Gates (X4000)

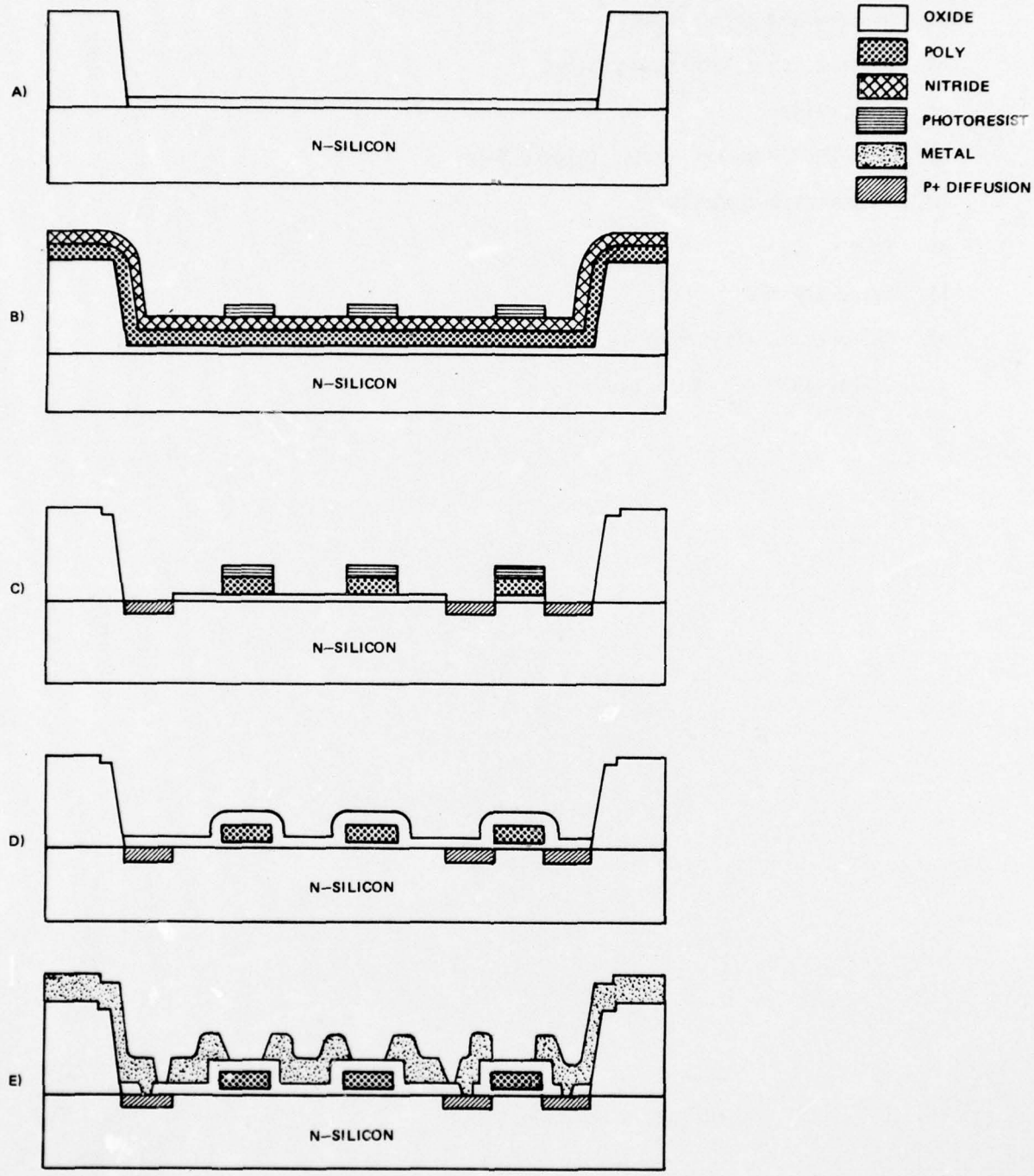


Figure 5-5. Steps in the DP1 Process

DP-1 processing continues as follows after step 16:

24. TEOS deposition; 725^{o}C in vacuum
25. Densify and getter TEOS
26. Photoresist polysilicon protect
27. Etch oxide
28. Grow 2000 Å channel oxide (Figure 5-5d)
29. Photoresist contacts
30. Etch oxide
31. Evaporate Al
32. Photoresist and define Al (Figure 5-5e)
33. Sinter 450^{o}C N_2 and H_2.

6. SIGNAL PROCESSING INTERFACE STUDIES

6.1 INTRODUCTION

6.1.1 Background

Networks of computers used for signal processing exhibit several unique features by comparison to networks of computers used for data processing or for realtime control (e.g., avionic data bus systems). The most obvious difference is that signal processing computers generally may be arranged to process data which is grouped into blocks, e.g., 256 words. This means that a substantial amount of time (1 to 10 msec), is required to transmit data between processors, and that a circuit switching interconnection scheme which requires, say, 10 to 100 μsec, to establish a data path is acceptable.

The second unique aspect of signal processing networks is the irregularity of typical network topology. The system designer needs freedom to adjust his system to the problem requirements. It is desirable to be able to connect the network of signal processors in the same flow form as the problem exhibits. An important aspect of this is the ability to optimize redundancy in each specific network. Redundancy is increased (at a cost of increased hardware complexity) by providing more than one signal path between important network resources. If any portion of one path fails, an alternate path still exists.

The third distinguishing aspect of signal processing networks is the frequent need to process data in a pipeline fashion. This arises because of the relative slowness of most signal processing computers with respect to the input data rate. The signal processing network designer is not permitted the luxury of transferring a single block of data at a time; many blocks must be transferred simultaneously within the system.

6.1.2 Overview

The network organization which evolved during this study is called the transparent interface system. In the taxonomy of Anderson and Jensen,* it is an irregular network of signal processors interconnected with dedicated paths using decentralized routing (Figure 6-1). Specifically, the network consists of an assortment of signal processing elements each of which is connected to a switching mode (or node processor). The switching nodes are interconnected with communication links.

*G.A. Anderson and E. Jensen, "Computer Interconnection Structures: Taxonomy, Characteristics and Examples," Computing Surveys, Vol. 7, December 1975, pp. 197-213.

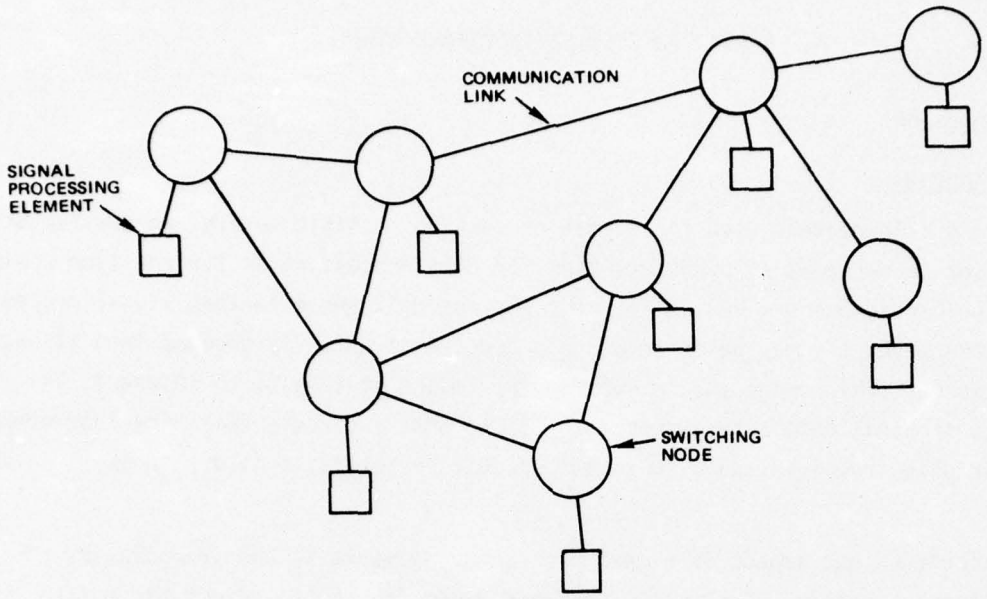


Figure 6-1. Network Organization

This study defines the processing required at the switching node. The node processor can be implemented in any of several LSI technologies depending on performance requirements and optimization criteria (size, power, development cost, reliability, etc.). The switching node may be implemented efficiently using the crosspoint switch and the path selection network described in detail in Section 6.4 of this report. Baseline nodes would have up to eight data ports, one of which connects to the "underlying" signal processor.

The only topological restriction imposed on the system designer is that only one link can be connected between any pair of nodes (i.e., links cannot be paralleled). In all other respects the designer enjoys complete freedom.

System control is distributed since each node contains an interconnection table which defines the complete system. This avoids the failure sensitivity of networks which have centralized control. Although the latter may be more efficient in terms of hardware, a single failure in the centralized controller can disable the entire network. The distributed control network exhibits desirable fail-soft characteristics; although a single failure may cripple a portion of the network, the remainder functions normally. Distributed control facilitates simultaneous utilization of many links. This is simplified by use of a nonhierarchical crosspoint switch at each node which allows multiple links to cross at each node.

Dynamic, associative priority is implemented; priority may change from message to message or within a message. If an interferer with sufficient priority requests a link which is in use, interrupts are sent to the link users informing them that the link must be relinquished within a time period determined by network protocol.

6.1.3 Summary

The preliminary system design is complete for the transparent interface system and is described in Section 6.2. The system has been simulated with a discrete event simulator as described in Section 6.3; the simulation was used to eliminate potential deadlock problems. Two LSI circuits have been designed as described in Section 6.4 to facilitate the implementation of efficient signal processing interconnection systems. The crosspoint switch circuit is of general applicability to all forms of circuit switching networks, while the path selection logic is optimized for this system. The transparent interface system presented here allows the signal processing designer to concentrate on solving the important signal processing problems — little effort need be expended on the interconnection problem which has been solved in an efficient manner which is directly suitable for LSI implementation.

6.2 DESCRIPTION OF TRANSPARENT INTERFACE SYSTEM

In this section a message routing algorithm is presented which can be used for any type of network topology ranging from in-line to fully matrixed. This allows the system designer great flexibility in choosing an interconnection scheme.

The topology of a network determines the amount of redundancy and failure sensitivity of the system. Redundancy is increased by adding alternate paths between the important nodes. Then, if an intermediate node along the primary path fails, or if one or more of the links in the primary path is already in use, an alternate path exists. Fault tolerance is introduced not only through the use of multiple paths, but also by distribution of control throughout the network. Thus the system does not require survivability of a central controller. Even if some nodes fail, the system will still be functional. Note, however, that certain communication paths may no longer be usable. Each node is independent; there is no common data structure or master controller.

Another feature of this system is that several communication paths can be used simultaneously. For example, a node can have several paths passing through it at once. This is due to the multiple bus circuit-switching technique which is used for routing information from source to destination.

Deadlock does not develop in this system because of the way in which the path selection algorithm is implemented. After a node controller has decided on a primary path for communication, it establishes the path by acquiring one link at a time. If any link is tied up, the node controller remembers which one it is, frees all the links that have been acquired, and formulates an alternate path. Deadlock would occur if a node controller were to hold onto the links it had previously acquired while waiting for the busy links it needs. In the proposed system if no alternate path exists, the node controller can either wait and retry the primary path, or increase the priority of its message and retry the path.

Flexibility is achieved by permitting a node controller to change its message priority at any time. Dynamic priority gives a node the ability to match priority to importance for different parts of a message. For example, if the message contains the status of a remote detection unit, some of the data will be of a housekeeping nature (i.e., temperature, etc.) which is not nearly as important as the data produced during the detection of a high priority event.

The assumed system ground rules for path selection are:

1. All communication links provide two-way communication.
2. A node can connect multiple paths simultaneously (i.e., cross bus).
3. The best path contains a minimum number of links.
4. If two or more paths contain the same number of links, either path is equally desirable.

Two data structures are used to select a path: the path matrix and threaded stack. The path matrix identifies all the interconnections between the nodes comprising the network (i.e., which nodes are interconnected). This matrix is resident in each node controller and may be updated during the path acquisition sequence. The threaded stack is used for the actual path selection. For any given node the stack is loaded by entering all those nodes adjacent to it, followed by those nodes which are two links away, etc. until the destination node is found. When a node name is entered in the stack, its connection to a prior node in the stack is indicated by the link field. The link field contains the stack address of the prior node. The path to the given node is found by threading back through the links to the originator.

6.2.1 Path Matrix Operations

The path matrix is the fundamental data structure for the entire system and therefore must be kept accurate at all times. If the network topology never changes, there is no need to burden the node controller with updating the matrix. In this case, the path matrix can be stored in a permanent read-only memory, such as a ROM. However, if the topology is expected to change due to node failures or addition and deletion of nodes, the path matrix must be stored in a read/write memory, such as a RAM. This section deals with two problems: creation of the path matrix upon power-up and regular updating of the path matrix to reflect recent network changes.

Suppose we have the network of Figure 6-2 and that we are node X4. Upon power-up we must determine the path matrix for this system. We will also assume that the node controller knows which port is active. Each node has a fixed maximum number of ports (e.g., eight), but not all of the ports will be used at any given time. For this reason, a status bit is associated with each port to indicate whether or not that port is currently being used. The node controller sends a "who are you?" command from each port to the adjacent nodes. These interrogated nodes will then reply with their name.

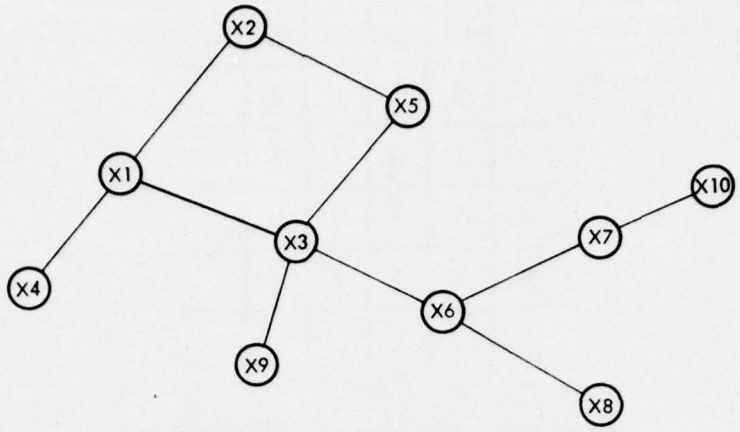


Figure 6-2. Network Topology

Thus, X4 sends "who are you?" through his only active port. This message is received and processed by X1 who responds with "I am X1." This information is now stored in matrix form as in Figure 6-3. A connection between two nodes is indicated by placing a "1" in the proper position. Notice that the matrix position (X4, X4) is zero due to the fact that X4 cannot talk to itself.

	X4	X1	
X4	0	1	
X1	?	?	

Figure 6-3. Path Matrix After Interrogating Neighbors

After a response has been received from each active port, the node controller interrogates each neighbor to determine their neighbors. Thus, X4 selects a path to X1 by executing the path selection algorithm and acquires the path to X1. After completing the path, X4 asks X1 "who are you connected to?" Node X1 will reply with a list of his nearest neighbors. In the event that X1 has not completed his list (e.g., upon simultaneous start-up of every node in the system), X1 will reply with a partial list response which provides X4 with X1's present information, but also warns X4 that X1 must be interrogated again later to complete that section of the path matrix.

Assuming that X1 has completed the list of his closest neighbors, he would respond to X4's question with "my neighbors are X4, X2, and X3." Node X4 would then add this new information to the path matrix which would result in the matrix of Figure 6-4.

	X4	X1	X2	X3	
X4	0	1	0	0	
X1	1	0	1	1	
X2	?	?	?	?	
X3	?	?	?	?	

Figure 6-4. Path Matrix After Asking X1 "Who Are Your Neighbors?"

This process of filling in vacant rows in the path matrix continues by forming a path to the node representing the next incomplete row in the matrix, which is X2. Once X4 has constructed a path to X2 he then asks "who are you connected to?" and X2 will respond with his list of neighbors. This information is inserted into the path matrix to yield a more complete picture of the network (Figure 6-5).

	X4	X1	X2	X3	X5	
X4	0	1	0	0	0	
X1	1	0	1	1	0	
X2	0	1	0	0	1	
X3	?	?	?	?	?	
X5	?	?	?	?	?	

Figure 6-5. Path Matrix After Building a Path to X2 and Asking Him, "Who Are Your Neighbors?"

The path matrix construction terminates when the lists of neighbors sent back from interrogated nodes contain no new node names. The final path matrix, from X4's point of view, is shown in Figure 6-6. Notice that since all communication links are bidirectional, the matrix has diagonal symmetry. If on the other hand, some of the links were unidirectional or if some line drivers have failed, the matrix would be unsymmetric. For example, if X4 could send data to X1, but not vice versa, the position (X4, X1) would be "1" while (X1, X4) would be "0".

	X4	X1	X2	X3	X5	X6	X9	X7	X8	X10	
X4	0	1	0	0	0	0	0	0	0	0	
X1	1	0	1	1	0	0	0	0	0	0	
X2	0	1	0	0	1	0	0	0	0	0	
X3	0	1	0	0	1	1	1	0	0	0	
X5	0	0	1	1	0	0	0	0	0	0	
X6	0	0	0	1	0	0	0	1	1	0	
X9	0	0	0	1	0	0	0	0	0	0	
X7	0	0	0	0	0	1	0	0	0	1	
X8	0	0	0	0	0	1	0	0	0	0	
X10	0	0	0	0	0	0	0	1	0	0	

Figure 6-6. Final Path Matrix

6.2.2 Path Selection

Path selection is a fairly straightforward procedure using the path matrix and threaded stack. Using the example network shown in Figure 6-2, we shall show how a path is created.

Assume that node X4 has a message for X5. By looking at the network topology we see that there are two paths to X5 from X4, both of which are equally desirable (since the assumed optimization criterion is to minimize the number of lines).

Each word on the path stack is divided into two parts: a node name and link back to a previous stack member, as shown in Figure 6-7. To initiate a path search, the first word placed on the stack contains the originator's name and a null link. Since every node can only be placed on the stack once, each node pushed is marked. After X4 has been marked, its row in the scratchpad path matrix (which is simply a copy of the "permanent" path matrix) is scanned for "1"s. Each time a "1" is found, the node associated with that column is pushed on the stack and linked to the node associated with the row which is presently being scanned, i.e., X4. The link concatenated with X1 will be X4's stack position (Figure 6-8).

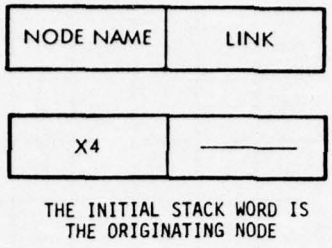


Figure 6-7. Stack Word

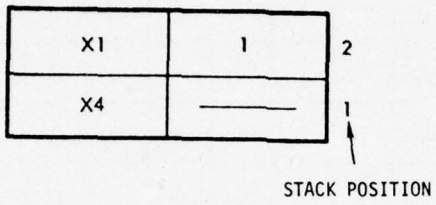


Figure 6-8. Stack After X4's Row in Path Matrix Has Been Connected

Since X1 is not the destination node, we must continue the search by putting any new candidates connected to X1 on the stack. So, scanning X1's row in the path matrix we find X1 is connected to X4, X2, and X3. But X4 has already been used, so only X2 and X3 are entered (Figure 6-9).

X3	2	4
X2	2	3
X1	1	2
4	—	1

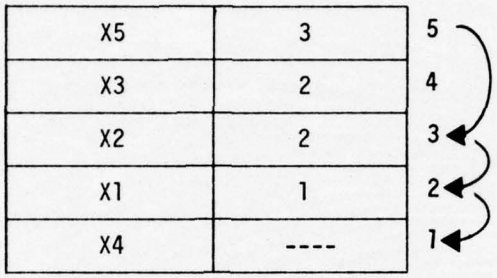
Figure 6-9. Stack After Scanning X1's Row in Path Matrix

The row belonging to the next point-of-search node in the stack, X2, must now be examined for nodal connections. From the scratchpad path matrix, X2 has connections to X3 and X5. Since node X3 has already been used, it is neglected and X5 is pushed on the stack and linked back to X2. We have finally found the destination node (Figure 6-10).

The path used to access this node is formulated by threading back through the linked list of nodes which uses the destination node as a list head. Thus, the path to X5 from X4 is X5-X2-X1-X4, as shown in Figure 6-11.

X5	3	5
X3	2	4
X2	2	3
X1	1	2
X4	———	1

Figure 6-10. Completed Stack



PROPOSED PRIMARY PATH: X5-X2-X1-X4

Figure 6-11. Path Threading

Now that a path has been proposed, X4 must acquire each link comprising the path one at a time. This is done by sending out "get link" (GTLNK) commands to the appropriate nodes. Initially X4 sends GTLNK, X1 to X1 since that is the first intermediate node in the path. Since the link is not currrently being used, X1 replies with a "link established" (LINKES). Having acquired the initial link, X4 now tries for the link connecting X1 and X2 by issuing a GTLNK, X2 to the channel connected to X1. Upon receiving the message, X1 first checks the status of the port leading to X2. If that port is free, X1 sends the original message to X2 and physically connects the port leading to X2 with the port leading to X4 using the crosspoint switch. Assuming X2 does not wish to use the link sought after by X4, he will respond with LINKES, which is passed through node X1 to X4.

Node X4 completes the path by sending a GTLNK, X5 along the unfinished path. This message passes to X2 which performs the check on the appropriate port and passes the message on to X5. If X5 responds with LINKES, then the path is complete. X4 must now send a request to use the processor interfaced to X5 (GETPRC). If X5's processor is busy the response will be PRCBSY; otherwise it will be PRCRDY.

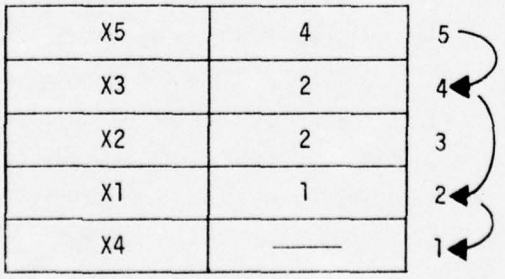
If a processor is busy, the originator sends a "broadcast: free link" (BFRLNK) command, which causes all the nodes along the completed path to relinquish the path segments. At this point, the node controller can do one of two things:

1) If another node is in the system with an underlying processor that performs the same function as processor node X5, a path to the alternate node can built.

2) If the processor at node X5 performs a unique function, the node controller must either throw the data away or keep trying X5 until he receives a PRCRDY.

In the above example we assumed that all the links requested by X4 were free. However, a desired link may be in use by another node controller who is sending a higher priority message. Let us assume that the response to GTLNK, X5 was "link busy" (LNKBSY) and that X4 does not wish to increase the priority of his message. Node X4 must enter the busy information into the scratchpad path matrix (since the link will be busy only temporarily) by setting the positions X2, X5, and X5, X2 to "0". The link already acquired (i.e., all acquired prior to receiving the LNKBSY) are released by issuing a BFRLNK. If the processor waited for the busy link to become free, a deadly embrace situation could occur.

X4 reruns the path selection algorithm with this new information injected into the scratchpad path matrix. Although most of the information left in the old stack is good, it is harder to delete the bad information than it is to start over again. The stack creation process remains the same up to the point in which X2's row is scanned. The result will now be that X2 is only connected to X3, but X3 has already been pushed on the stack. Thus, X2 is a dead end and X3's row is examined for possible links. The first unused connection in X3's row is the X3, X5 position. Once again the destination has been reached, and a new path is proposed as shown in Figure 6-12.



PROPOSED ALTERNATE PATH: X5-X3-X1-X4

Figure 6-12. Threading Alternate Stack

X4 again tries to establish the new proposed path. If the new path cannot be completed, X4 marks the busy link in the scratchpad path matrix and tries once more. X4 keeps trying until it is not possible to reach the destination node. X5 can then wait and try again from the beginning, or raise the message priority.

6.2.3 Flowcharts

The algorithm implemented for a path matrix construction is shown in Figure 6-13. It is assumed that all matrix locations contain a zero initially since it is assumed that there will be fewer connections than nonconnections.

The matrix updating process is merely a subset of the path matrix construction procedure. If X4 is suspicious of the condition of some node in the system, he can

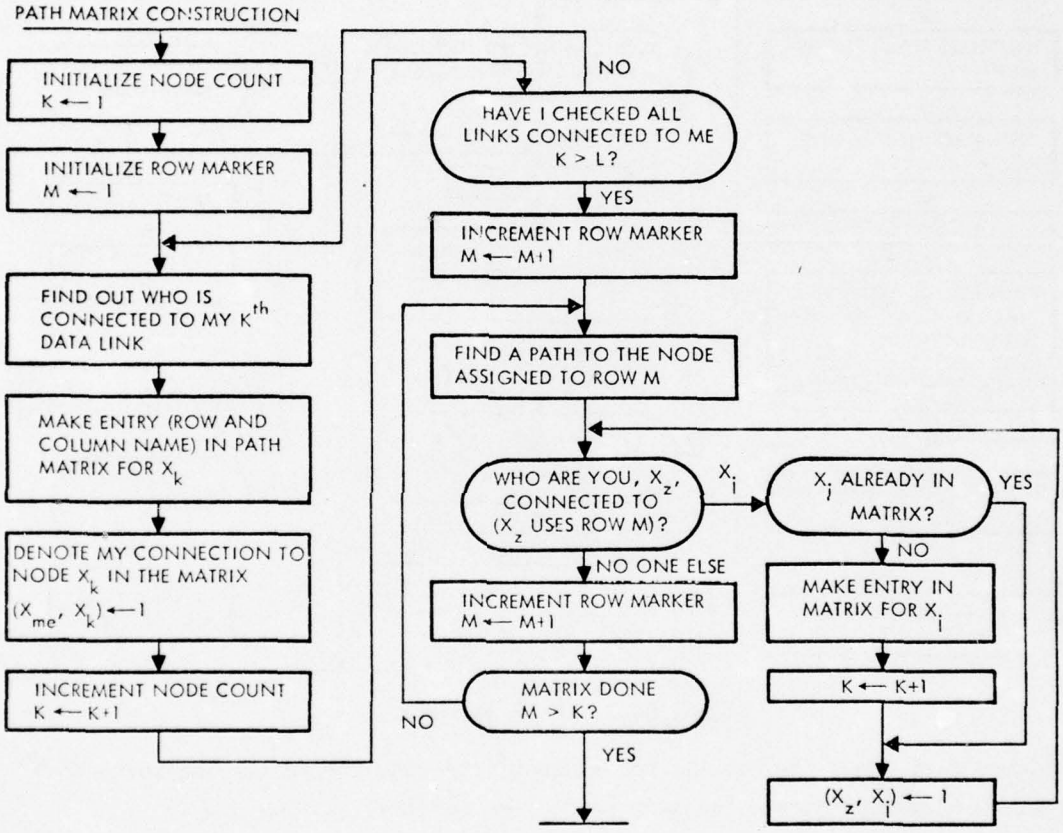
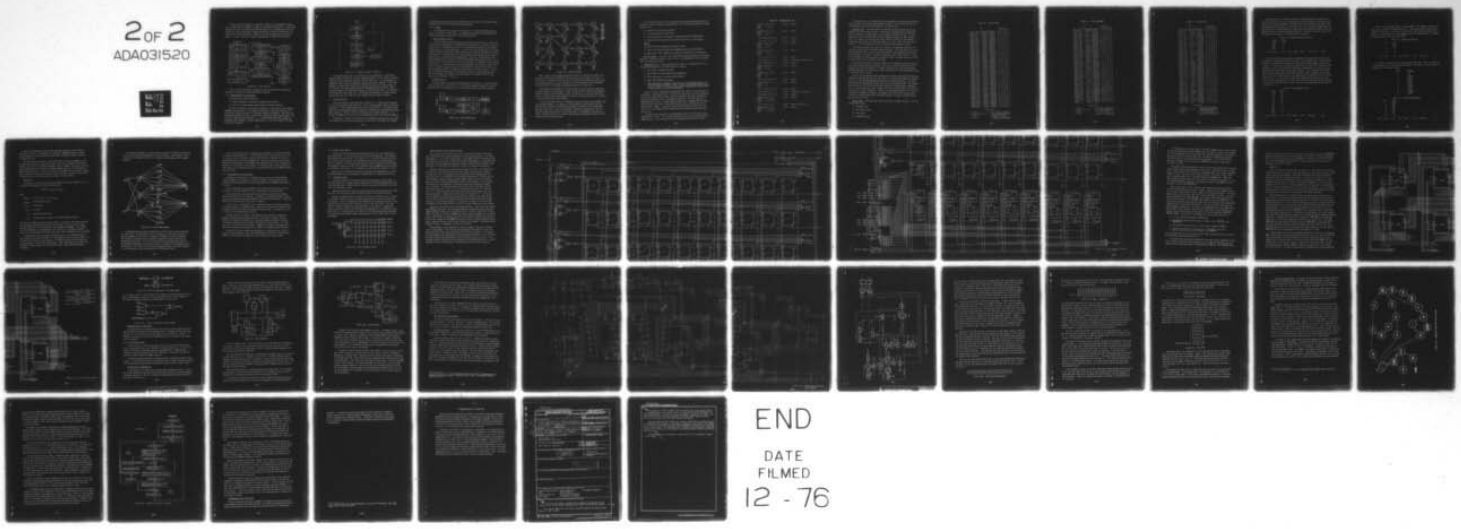


Figure 6-13. Path Matrix Construction

send out a "who are you connected to?" to the node in question and use the response to make any necessary changes in the path matrix. If the network has a dynamic topology, all the nodes in the system must periodically send out interrogations to all other nodes in the system to discover additions or deletions to the network. If it is undesirable to have the nodes constantly updating their matrices, then one node in the system, acting as a monitor, could inform all other nodes that a change has been made to the system and that they should refresh their matrices. In this way, the path matrix updating process would be executed only when there is a change in the network topology.

AD-A031 520 TRW SYSTEMS GROUP REDONDO BEACH CALIF F/G 9/5
CHARGE COUPLED DEVICES IN SIGNAL PROCESSING SYSTEMS. VOLUME III--ETC(U)
MAR 76 T A ZIMMERMAN, R J HANDY, R A ALLEN N00014-74-C-0068
UNCLASSIFIED NL

2 OF 2
ADA031520



END

DATE
FILMED
12 - 76

The path selection algorithm is presented in detail in the flowchart of Figure 6-14. Notice that the flowchart is exited if no connection is possible. At this time, the node controller could keep trying the same node, try a different node, raise the priority, or reject the message. Another possibility is that the desired node is no longer active in the system due to failure or deliberate deletion of a node in the system. If that is the case, it would be appropriate to inform the other nodes.

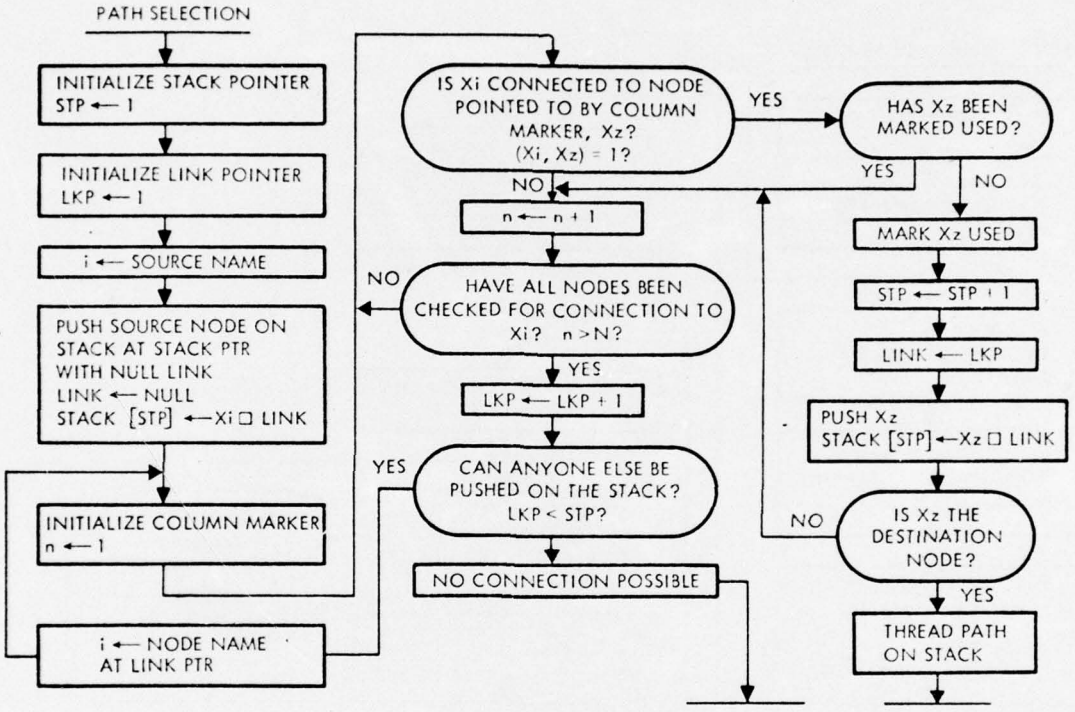


Figure 6-14. Path Selection

Figure 6-15 shows the overall operation of the node controller including the path matrix construction and the path selection routines.

6.2.4 Recommendations for Future Work

Two problem areas remain unresolved:

1) Startup procedures for systems with dynamic path matrices

2) Protocols for the processor which "underly" the node controller.

For the first problem, it was noted in Section 6.2.1 that partial lists might be exchanged during startup. If all nodes are powered-up at about the same time it is unclear that the path matrices will converge to complete information. This should be investigated analytically prior to use of this scheme with dynamic path matrices. Use of fixed or semifixed path matrices (i.e., a single network controller reloads all path matrix RAM's whenever a network change occurs) is a solution; however, the question remains unanswered.

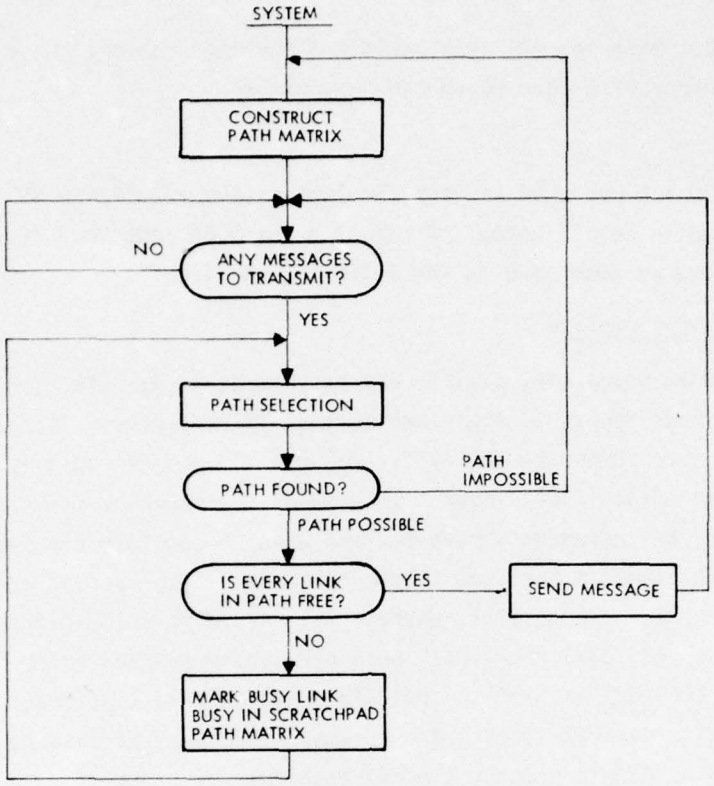


Figure 6-15. Flowchart of Overall Operation

The second problem leads to the question of what information an "underlying" processor needs to effectively use this networking scheme. At one extreme a processor might specify the destination processor for each data packet. If a path cannot be completed a simple "path not completed" message could be the response. At the other extreme a processor might simply denote the type of processor to which the data is to be routed. The latter approach could be implemented easily by adding a simple processor between the "underlying" processor and the node controller. The same effect can be obtained in the former case if the underlying processor is a general purpose machine with sufficient reserve to add a network resource selection program.

6.3 SIMULATION RESULTS

Using the interface system described in Section 6.2, a simulation was conducted to reveal the operational characteristics, and to indicate whether deadlock situations might arise. A simple SONAR problem was used for the simulation. The simulation was performed in SALSIM (System Analysis Language for SIMulation), a TRW developed discrete event simulator. SALSIM is similar to the well-known simulator Simscript. It is written in FORTRAN IV and runs on the TRW timesharing system using a CDC 6600 computer.

As described in detail (in Section 6.3.1) no deadlocks occurred within the system. The overhead due to the interface system was on the order of 32 percent even though

the network had not been specifically optimized for the problem. With minor changes overhead on the order of 5 percent should be achieved.

6.3.1 Example

The problem which was used for the simulation is a simplified SONAR processor, which is described in detail below. A solution using 20 nodes and 30 communication links was simulated as described in the following section.

SONAR System Description

The basic SONAR processing example may be understood by referring to Figure 6-16. Data acquired by hydrophones is digitized by the A/D converters, filtered through the digital filters, transformed by the FFT's, accumulated and averaged in the ensemble averaging unit, and finally displayed. In view of the block orientation of our system, the A/D converters are assumed to have buffers which accumulate 256 samples per hydrophone channel. The samples here and in the remainder of the system are 32-bit complex words (16 bits in-phase and 16 bits quadrature). Assuming a 128 hydrophone system (64 hydrophones per A/D converter) with each hydrophone sampled every 0.4 msec (at the Nyquist rate for 1250 Hz) each of the two A/D converters will generate a block of data every 1.6 msec. The digital filters are assumed capable of processing a block of data in 25.6 μsec (100 nsec/data point), the FFT transforms a block of data in 200 μsec, and the ensemble averager is assumed to require 256 μsec per block of data. The display is assumed to have access to the ensembles of data on a noninterfering basis. The final parameter of interest is the time required to transmit a block of data from one unit to another – 256 μsec.

Simulated System

A block diagram of the simulated system appears in Figure 6-17. Twenty processors (two A/D input units, three filters, 13 FFTs, and two ensemble averagers) are interconnected with 30 transmission links. No attempt was made to optimize the network.

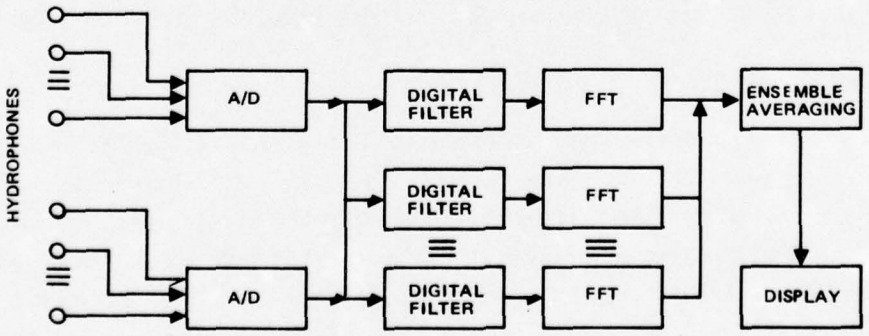


Figure 6-16. Basic SONAR System

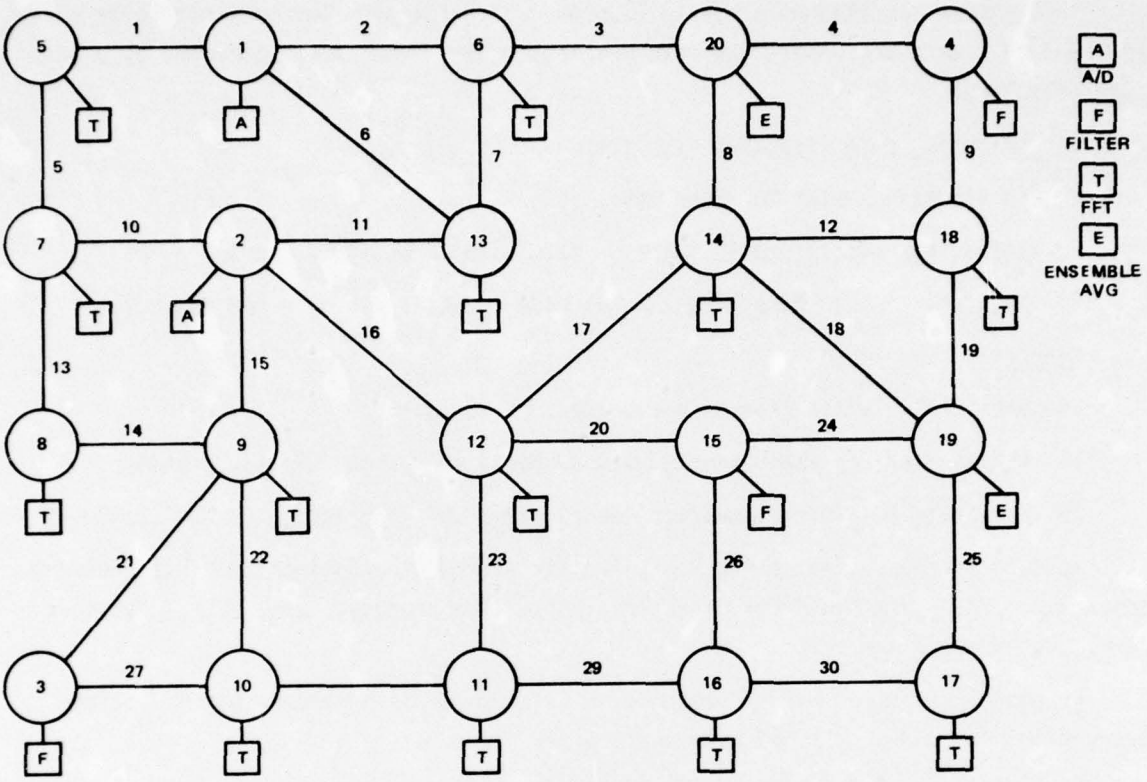


Figure 6-17. Simulated System

When a block of data becomes available at either A/D converter (as noted in the previous section, each A/D generates a block of data every 1.6 msec) it is sent to two filters, or alternatively to the same filter twice; this allows each block of data to be processed with two cutoff frequencies. The filtered data is then transformed and finally sent to the ensemble averager which serves as a data sink.

Each processor was assumed to be single buffered so that at most one data block could be resident at each processor. Thus, the basic steps in the flow of information through a node are to receive data, process it, and send it to the next appropriate node. If a node is busy processing when a request to send more data to it is received, it replies with processor busy (PRCBSY).

If a multiple buffer scheme is implemented at each processor, the throughput time will decrease somewhat. For example, if each processor has two buffers, then one buffer can be filling while the processor operates on the data in the second buffer. When the input buffer is full and processing is complete, processing will start on the input buffer while the data in the other buffer is transmitted to the next processor. Double buffering is of greatest benefit when the transmission time is of the same order as the processing time.

This simulation assumes a static topology which relieves the node controllers of the need to create and update the path matrices. Thus, the tasks performed by a node controller are:

1) Executing path selection algorithm

2) Constructing paths to other nodes

3) Servicing requests to initiate or terminate connections between ports

4) Keeping track of busy ports at the node and busy links in the network.

Results

The SALSIM simulation generates two types of output:

1) Action listings which report what happened at various simulated times

2) Result tables which summarize the observed characteristics of the system.

Action Listings. The action lists facilitiate minute examination of the network operation. There are two forms of action lists: the proposed path file and various forms of message trace lists.

Proposed Path File. The proposed path file, which is a chronological list of every path proposed by all the node controllers, reports:

1) Which node is proposing the path

2) Which nodes and links comprise the proposed path

3) Which data block is to be sent along the path

4) What time the path was selected

5) Which data block is presently using the links in the proposed path (if the links are free then the user is "0"). The list also reports the node controller's success or failure at completing the proposed path and the reason for the failure (e.g., which link was busy).

The first entry in the example proposed path file (Table 6-1) will be used for explanation. The proposed path is to be used for data block 1, and was selected at 10 μsec after system startup (t=0). The path originates from node 1 and leads to node 4 using nodes 6 and 20 as intermediate connections. The links used are 2 (between nodes 1 and 6), 3 (between nodes 6 and 20), and 4 (between nodes 20 and 4). None of these links is busy at this time (10 μsec) which is indicated by the "0" under each link number. If some data block had been present on any of the links (2, 3, or 4), its number would have been placed in the USERS row under the appropriate link.

The fourth entry shows that the path proposed in the first entry was finally established at 25 μsec. Thus, the node controller took 15 μsec to acquire the proposed path and to receive permission to use the underlying processor at node 4.

Table 6-1. Proposed Path File

```
 PROPOSED PATH FOR ID          1 TIME=   .000010
NODES   1  6 20  4
LINKS      2  3  4
USERS      0  0  0

 PROPOSED PATH FOR ID          3 TIME=   .000010
NODES   2  9  3
LINKS     15 21
USERS      0  0

 PATH ESTABLISHED FOR ID       3 TIME=   .000019

 PATH ESTABLISHED FOR ID       1 TIME=   .000025

 PROPOSED PATH FOR ID          4 TIME=   .000289
NODES   2  9  3
LINKS     15 21
USERS      0  0

 PROPOSED PATH FOR ID          2 TIME=   .000294
NODES   1  6 20  4
LINKS      2  3  4
USERS      0  0  0

 PATH ESTABLISHED FOR ID       4 TIME=   .000298

 PATH FAILURE FOR ID           4 TIME=   .000300 BUSY PROCESSOR=   3

 PATH ESTABLISHED FOR ID       2 TIME=   .000309

 PROPOSED PATH FOR ID          4 TIME=   .000312
NODES   2 12 15
LINKS     16 20
USERS      0  0

 PROPOSED PATH FOR ID          3 TIME=   .000314
NODES   3  9
LINKS     21
USERS      0

 PATH ESTABLISHED FOR ID       3 TIME=   .000318

 PROPOSED PATH FOR ID          1 TIME=   .000319
NODES   4 18
LINKS      9
USERS      0

 PATH ESTABLISHED FOR ID       4 TIME=   .000320

 PATH FAILURE FOR ID           2 TIME=   .000322 BUSY PROCESSOR=   4

 PATH ESTABLISHED FOR ID       1 TIME=   .000323

 PROPOSED PATH FOR ID          2 TIME=   .000334
NODES   1 13  2  9  3
LINKS      6 11 15 21
USERS      0  0  0  3

 PATH FAILURE FOR ID           2 TIME=   .000349 BUSY LINE=  21

 PROPOSED PATH FOR ID          2 TIME=   .000362
NODES   1 13  2 12 15
LINKS      6 11 16 20
USERS      0  0  4  4
```

The eighth entry in the proposed path file indicates a failure by node controller 2 to gain access to the underlying processor at node 3. All the links for the proposed path were acquired but the processor is already in use.

Message Traces. The other method for examining the detailed operation of the network is to use the message traces. Three types of message traces are generated by this simulation: trace by node, trace by message, and trace by time. All the message traces have the same format but are grouped according to a node, message, or time. For instance, the trace by node chart (Table 6-2) groups all the messages transmitted and received at a particular node, e.g., node 1, and sorts them chronologically. Thus, the message activity of any node can be examined in detail.

The trace by message chart (Table 6-3) groups all the messages associated with a particular data block as it moves through the system. Thus, the progress of a data block generated at node 1 (one of the A/D converters) at the beginning of simulation can be followed in the net as it moves from the source to a filter and so on. Also, all the commands and responses used to generate the path (including the partial paths) for each data block are included in the trace.

The trace by time chart (Table 6-4) groups all the messages in the system according to the time which they were created. This trace is used to examine the overall interaction of the node controllers. Network characteristics, such as bottlenecks and deadly embraces, can be seen quite easily by looking at all the messages generated in the system at any instant.

Each line of the message traces contains the simulation time when the message was created, the message type, the data block number, the function and number of the node of interest, and some housekeeping data used by the simulation program. The fourth column of the housekeeping data on the chart lists the destination node. For example, the second entry in the trace by node chart shows that a GTLNK message originates at node 1, a source node, that it was created at t=10 μsec for data block 1, and that it is being sent to node 6 (column 4). Column 1 of the housekeeping data lists the node from which the response originated. For example, entry four of the trace by node listing is a LINKES response sent from node 6 (column 1) to node 1 (column 4) which was received at t=14 μsec.

Result Tables. The simulation results are given in a number of tables. The four most interesting are:

1) Throughput time
2) Path establish time
3) Path length
4) Transmission type.

Table 6-2. Trace by Node

TIME	MESSAGE	DATA BLOCK	FUNCTION	NODE	FUNCTION	HOUSEKEEPING 1	2	3	4	5	6
0.000000	STX	1	SOURCE	1	OUTPUT	0	1	0	0	0	1
.000010	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000012	STX	2	SOURCE	1	OUTPT QUE	0	1	0	0	0	1
.000014	LINKES	1	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000014	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000019	LINKES	1	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000019	GTLNK	1	SOURCE	1	TRANSMIT	1	1	20	4	4	2
.000025	LINKES	1	SOURCE	1	PROCESSOR	4	20	1	1	4	2
.000025	GETPRC	1	SOURCE	1	TRANSMIT	1	1	4	0	0	3
.000026	PRCRDY	1	SOURCE	1	PROCESSOR	4	4	1	0	0	3
.000026	STX	1	SOURCE	1	TRANSMIT	0	1	4	0	0	3
.000284	ENDXMT	1	SOURCE	1	PROCESSOR	0	0	0	0	0	1
.000284	BFRLNK	1	SOURCE	1	TRANSMIT	1	1	6	4	2	1
.000284	STX	2	SOURCE	1	OUTPUT	0	1	0	0	0	1
.000294	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000298	LINKES	2	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000298	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000303	LINKES	2	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000303	GTLNK	2	SOURCE	1	TRANSMIT	1	1	20	4	4	2
.000309	LINKES	2	SOURCE	1	PROCESSOR	4	20	1	1	4	2
.000309	GETPRC	2	SOURCE	1	TRANSMIT	1	1	4	0	0	3
.000322	PRCBSY	2	SOURCE	1	PROCESSOR	4	4	1	0	0	3
.000322	BFRLNK	2	SOURCE	1	TRANSMIT	1	1	6	4	0	1
.000334	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	13	6	1
.000337	LINKES	2	SOURCE	1	PROCESSOR	13	13	1	1	6	1
.000337	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	2	11	1
.000342	LINKES	2	SOURCE	1	PROCESSOR	2	13	1	1	11	1
.000342	GTLNK	2	SOURCE	1	TRANSMIT	1	1	2	9	15	2
.000347	LINKES	2	SOURCE	1	PROCESSOR	9	2	1	1	15	2
.000347	GTLNK	2	SOURCE	1	TRANSMIT	1	1	9	3	21	3
.000349	LNKBSY	2	SOURCE	1	PROCESSOR	9	9	1	1	21	1
.000349	BFRLNK	2	SOURCE	1	TRANSMIT	1	1	13	9	0	1
.000362	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	13	6	1
.000365	LINKES	2	SOURCE	1	PROCESSOR	13	13	1	1	6	1
.000365	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	2	11	1
.000370	LINKES	2	SOURCE	1	PROCESSOR	2	13	1	1	11	1
.000370	GTLNK	2	SOURCE	1	TRANSMIT	1	1	2	12	16	2
.000373	LNKBSY	2	SOURCE	1	PROCESSOR	2	2	1	1	16	1
.000373	BFRLNK	2	SOURCE	1	TRANSMIT	1	1	13	2	0	1
.000385	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000388	LINKES	2	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000388	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000393	LINKES	2	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000393	GTLNK	2	SOURCE	1	TRANSMIT	1	1	20	14	8	2
.000398	LINKES	2	SOURCE	1	PROCESSOR	14	20	1	1	8	2
.000398	GTLNK	2	SOURCE	1	TRANSMIT	1	1	14	12	17	3
.000403	LINKES	2	SOURCE	1	PROCESSOR	12	14	1	1	17	3
.000403	GTLNK	2	SOURCE	1	TRANSMIT	1	1	12	15	20	4
.000406	LNKBSY	2	SOURCE	1	PROCESSOR	12	12	1	1	20	1
.000406	BFRLNK	2	SOURCE	1	TRANSMIT	1	1	6	12	0	1
.000419	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000422	LINKES	2	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000422	GTLNK	2	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000427	LINKES	2	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000427	GTLNK	2	SOURCE	1	TRANSMIT	1	1	20	14	8	2
.000432	LINKES	2	SOURCE	1	PROCESSOR	14	20	1	1	6	2
.000432	GTLNK	2	SOURCE	1	TRANSMIT	1	1	14	19	18	3
.000437	LINKES	2	SOURCE	1	PROCESSOR	19	14	1	1	18	3
.000437	GTLNK	2	SOURCE	1	TRANSMIT	1	1	19	15	24	4
.000442	LINKES	2	SOURCE	1	PROCESSOR	15	19	1	1	24	4
.000442	GETPRC	2	SOURCE	1	TRANSMIT	1	1	15	0	0	5
.000444	PRCBSY	2	SOURCE	1	PROCESSOR	15	15	1	0	0	5
.000444	BFRLNK*	2	SOURCE	1	TRANSMIT	1	1	6	15	0	1
.000456	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	13	6	1
.000459	LINKES	2	SOURCE	1	PROCESSOR	13	13	1	1	6	1
.000459	GTLNK	2	SOURCE	1	TRANSMIT	1	1	13	2	11	1

Housekeeping Data

1. Originator
2. Intermediate node (if any)
3. Receiver
4. Destination
5. The link sought by a GETLNK command, the link established by a LINKES command, or the link that was unavailable (LNKBSY).
6. The number of links between the intermediate node and the receiver.

Table 6-3. Trace by Message

TIME	MESSAGE	DATA BLOCK	FUNCTION	NODE	FUNCTION	HOUSEKEEPING 1	2	3	4	5	6
0.000000	STX	1	SOURCE	1	OUTPUT	0	1	0	0	0	1
.000010	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000013	GTLNK	1	FFT	6	PROCESSOR	1	1	6	6	2	1
.000013	LINKES	1	FFT	6	TRANSMIT	6	6	1	1	2	1
.000014	LINKES	1	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000014	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000015	GTLNK	1	FFT	6	PROCESSOR	1	1	6	20	3	1
.000015	GTLNK	1	FFT	6	TRANSMIT	1	6	20	20	3	1
.000017	GTLNK	1	INTEGR	20	PROCESSOR	1	6	20	20	3	1
.000017	LINKES	1	INTEGR	20	TRANSMIT	20	20	6	1	3	1
.000018	LINKES	1	FFT	6	PROCESSOR	20	20	6	1	3	1
.000018	LINKES	1	FFT	6	TRANSMIT	20	6	1	1	3	1
.000019	LINKES	1	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000019	GTLNK	1	SOURCE	1	TRANSMIT	1	1	20	4	4	2
.000020	GTLNK	1	INTEGR	20	PROCESSOR	1	1	20	4	4	2
.000020	GTLNK	1	INTEGR	20	TRANSMIT	1	20	4	4	4	1
.000022	GTLNK	1	FILTER	4	PROCESSOR	1	20	4	4	4	1
.000022	LINKES	1	FILTER	4	TRANSMIT	4	4	20	1	4	1
.000024	LINKES	1	INTEGR	20	PROCESSOR	4	4	20	1	4	1
.000024	LINKES	1	INTEGR	20	TRANSMIT	4	20	1	1	4	2
.000025	LINKES	1	SOURCE	1	PROCESSOR	4	20	1	1	4	2
.000025	GETPRC	1	SOURCE	1	TRANSMIT	1	1	4	0	0	3
.000025	GETPRC	1	FILTER	4	PROCESSOR	1	1	4	0	0	3
.000025	PRCRDY	1	FILTER	4	TRANSMIT	4	4	1	0	0	3
.000026	PRCRDY	1	SOURCE	1	PROCESSOR	4	4	1	0	0	3
.000026	STX	1	SOURCE	1	TRANSMIT	0	1	4	0	0	3
.000284	STX	1	FILTER	4	INPUT	0	1	4	0	0	3
.000284	ENDXMT	1	SOURCE	1	PROCESSOR	0	0	0	0	0	1
.000284	BFRLNK	1	SOURCE	1	TRANSMIT	1	1	6	4	2	1
.000284	BFRLNK	1	FFT	6	PROCESSOR	1	1	6	4	2	1
.000284	BFRLNK	1	FFT	6	TRANSMIT	1	6	20	4	2	1
.000286	BFRLNK	1	INTEGR	20	PROCESSOR	1	6	20	4	2	1
.000286	BFRLNK	1	INTEGR	20	TRANSMIT	1	20	4	4	2	1
.000287	BFRLNK	1	FILTER	4	PROCESSOR	1	20	4	4	2	1
.000309	STX	1	FILTER	4	OUTPUT	0	4	0	0	0	3
.000319	GTLNK	1	FILTER	4	TRANSMIT	4	4	18	18	9	1
.000322	GTLNK	1	FFT	18	PROCESSOR	4	4	18	18	9	1
.000322	LINKES	1	FFT	18	TRANSMIT	18	18	4	4	9	1
.000323	LINKES	1	FILTER	4	PROCESSOR	18	18	4	4	9	1
.000323	GETPRC	1	FILTER	4	TRANSMIT	4	4	18	0	0	1
.000324	GETPRC	1	FFT	18	PROCESSOR	4	4	18	0	0	1
.000324	PRCRDY	1	FFT	18	TRANSMIT	18	18	4	0	0	1
.000325	PRCRDY	1	FILTER	4	PROCESSOR	18	18	4	0	0	1
.000325	STX	1	FILTER	4	TRANSMIT	0	4	18	0	0	1
.000582	STX	1	FFT	18	INPUT	0	4	18	0	0	1
.000582	ENDXMT	1	FILTER	4	PROCESSOR	0	0	0	0	0	1
.000582	BFRLNK	1	FILTER	4	TRANSMIT	4	4	18	18	9	1
.000583	BFRLNK	1	FFT	18	PROCESSOR	4	4	18	18	9	1
.000782	STX	1	FFT	18	OUTPUT	0	18	0	0	0	1
.000793	GTLNK	1	FFT	18	TRANSMIT	18	18	19	19	19	1
.000795	GTLNK	1	INTEGR	19	PROCESSOR	18	18	19	19	19	1
.000795	LINKES	1	INTEGR	19	TRANSMIT	19	19	18	18	19	1
.000797	LINKES	1	FFT	18	PROCESSOR	19	19	18	18	19	1
.000797	GETPRC	1	FFT	18	TRANSMIT	18	18	19	0	0	1
.000797	GETPRC	1	INTEGR	19	PROCESSOR	18	18	19	0	0	1
.000797	PRCRDY	1	INTEGR	19	TRANSMIT	19	19	18	0	0	1
.000798	PRCRDY	1	FFT	18	PROCESSOR	19	19	18	0	0	1
.000798	STX	1	FFT	18	TRANSMIT	0	18	19	0	0	1
.001056	STX	1	INTEGR	19	INPUT	0	18	19	0	0	1
.001056	ENDXMT	1	FFT	18	PROCESSOR	0	0	0	0	0	1
.001056	BFRLNK	1	FFT	18	TRANSMIT	18	18	19	19	19	1
.001056	BFRLNK	1	INTEGR	19	PROCESSOR	18	18	19	19	19	1
.000012	STX	2	SOURCE	1	OUTPT QUE	0	1	0	0	0	1
.000284	STX	2	SOURCE	1	OUTPUT	0	1	0	0	0	1

Housekeeping Data

1. Originator
2. Intermediate node (if any)
3. Receiver
4. Destination
5. The link sought by a GETLNK command, the link established by a LINKES command, or the link that was unavailable (LNKBSY).
6. The number of links between the intermediate node and the receiver.

Table 6-4. Trace by Time

TIME	MESSAGE	DATA BLOCK	FUNCTION	NODE	FUNCTION	HOUSEKEEPING 1	2	3	4	5	6
0.000000	STX	1	SOURCE	1	OUTPUT	0	1	0	0	0	1
0.000000	STX	3	SOURCE	2	OUTPUT	0	2	0	0	0	1
.000010	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	6	2	1
.000010	GTLNK	3	SOURCE	2	TRANSMIT	2	2	9	9	15	1
.000012	STX	2	SOURCE	1	OUTPT QUE	0	1	0	0	0	1
.000012	STX	4	SOURCE	2	OUTPT QUE	0	2	0	0	0	1
.000013	GTLNK	1	FFT	6	PROCESSOR	1	1	6	6	2	1
.000013	LINKES	1	FFT	6	TRANSMIT	6	6	1	1	2	1
.000013	GTLNK	3	FFT	9	PROCESSOR	2	2	9	9	15	1
.000013	LINKES	3	FFT	9	TRANSMIT	9	9	2	2	15	1
.000014	LINKES	1	SOURCE	1	PROCESSOR	6	6	1	1	2	1
.000014	GTLNK	1	SOURCE	1	TRANSMIT	1	1	6	20	3	1
.000014	LINKES	3	SOURCE	2	PROCESSOR	9	9	2	2	15	1
.000014	GTLNK	3	SOURCE	2	TRANSMIT	2	2	9	3	21	1
.000015	GTLNK	1	FFT	6	PROCESSOR	1	1	6	20	3	1
.000015	GTLNK	1	FFT	6	TRANSMIT	1	6	20	20	3	1
.000015	GTLNK	3	FFT	9	PROCESSOR	2	2	9	3	21	1
.000015	GTLNK	3	FFT	9	TRANSMIT	2	9	3	3	21	1
.000017	GTLNK	1	INTEGR	20	PROCESSOR	1	6	20	20	3	1
.000017	LINKES	1	INTEGR	20	TRANSMIT	20	20	6	1	3	1
.000017	GTLNK	3	FILTER	3	PROCESSOR	2	9	3	3	21	1
.000017	LINKES	3	FILTER	3	TRANSMIT	3	3	9	2	21	1
.000018	LINKES	1	FFT	6	PROCESSOR	20	20	6	1	3	1
.000018	LINKES	1	FFT	6	TRANSMIT	20	6	1	1	3	1
.000018	LINKES	3	FFT	9	PROCESSOR	3	3	9	2	21	1
.000018	LINKES	3	FFT	9	TRANSMIT	3	9	2	2	21	1
.000019	LINKES	1	SOURCE	1	PROCESSOR	20	6	1	1	3	1
.000019	GTLNK	1	SOURCE	1	TRANSMIT	1	1	20	4	4	2
.000019	LINKES	3	SOURCE	2	PROCESSOR	3	9	2	2	21	1
.000019	GETPRC	3	SOURCE	2	TRANSMIT	2	2	3	0	0	2
.000020	GETPRC	3	FILTER	3	PROCESSOR	2	2	3	0	0	2
.000020	PRCRDY	3	FILTER	3	TRANSMIT	3	3	2	0	0	2
.000020	GTLNK	1	INTEGR	20	PROCESSOR	1	1	20	4	4	2
.000020	GTLNK	1	INTEGR	20	TRANSMIT	1	20	4	4	4	1
.000021	PRCRDY	3	SOURCE	2	PROCESSOR	3	3	2	0	0	2
.000021	STX	3	SOURCE	2	TRANSMIT	0	2	3	0	0	2
.000022	GTLNK	1	FILTER	4	PROCESSOR	1	20	4	4	4	1
.000022	LINKES	1	FILTER	4	TRANSMIT	4	4	20	1	4	1
.000024	LINKES	1	INTEGR	20	PROCESSOR	4	4	20	1	4	1
.000024	LINKES	1	INTEGR	20	TRANSMIT	4	20	1	1	4	2
.000025	LINKES	1	SOURCE	1	PROCESSOR	4	20	1	1	4	2
.000025	GETPRC	1	SOURCE	1	TRANSMIT	1	1	4	0	0	3
.000025	GETPRC	1	FILTER	4	PROCESSOR	1	1	4	0	0	3
.000025	PRCRDY	1	FILTER	4	TRANSMIT	4	4	1	0	0	3
.000026	PRCRDY	1	SOURCE	1	PROCESSOR	4	4	1	0	0	3
.000026	STX	1	SOURCE	1	TRANSMIT	0	1	4	0	0	3
.000278	STX	3	FILTER	3	INPUT	0	2	3	0	0	2
.000278	ENDXMT	3	SOURCE	2	PROCESSOR	0	0	0	0	0	1
.000278	BFRLNK	3	SOURCE	2	TRANSMIT	2	2	9	3	15	1
.000279	STX	4	SOURCE	2	OUTPUT	0	2	0	0	0	1
.000279	BFRLNK	3	FFT	9	PROCESSOR	2	2	9	3	15	1
.000279	BFRLNK	3	FFT	9	TRANSMIT	2	9	3	3	15	1
.000280	BFRLNK	3	FILTER	3	PROCESSOR	2	9	3	3	15	1
.000284	STX	1	FILTER	4	INPUT	0	1	4	0	0	3
.000284	ENDXMT	1	SOURCE	1	PROCESSOR	0	0	0	0	0	1
.000284	BFRLNK	1	SOURCE	1	TRANSMIT	1	1	6	4	2	1
.000284	STX	2	SOURCE	1	OUTPUT	0	1	0	0	0	1
.000284	BFRLNK	1	FFT	6	PROCESSOR	1	1	6	4	2	1
.000294	BFRLNK	1	FFT	6	TRANSMIT	1	6	20	4	2	1
.000296	BFRLNK	1	INTEGR	20	PROCESSOR	1	6	20	4	2	1
.000296	BFRLNK	1	INTEGR	20	TRANSMIT	1	20	4	4	2	1
.000287	BFRLNK	1	FILTER	4	PROCESSOR	1	20	4	4	2	1
.000289	GTLNK	4	SOURCE	2	TRANSMIT	2	2	9	9	15	1
.000291	GTLNK	4	FFT	9	PROCESSOR	2	2	9	9	15	1

Housekeeping Data

1. Originator
2. Intermediate node (if any)
3. Receiver
4. Destination
5. The link sought by a GETLNK command, the link established by a LINKES command, or the link that was unavailable (LNKBSY).
6. The number of links between the intermediate node and the receiver.

The throughput time is defined as the total time which elapses from creation of the data block at the source until completion of the transmission of the data block to either of the ensemble averagers. The TIME column on the throughput time table (Table 6-5) quantizes the throughput times into 500 μsec or less; notice for example that 10 data blocks were processed through the system with times ranging from 1000 to 1500 μsec. When the expected value and standard deviation are calculated, the exact values of the throughput time (instead of the quantized values) are used.

Table 6-5. Throughput Time Distribution

TIME MICSEC	FREQ-UENCY
500	0
1000	0
1500	10
2000	5
2500	5

TOTAL SAMPLES 20 EXP. X VALUE 1649.1 STD. DEV. 301.4

The time taken by a node controller to establish a proposed path is affected by two factors: path length and node utilization in the proposed paths. Node controllers that must service many link requests will take longer, on the average, to respond to a request. As can be seen from the path establish time (Table 6-6) no paths were completed in 10 μsec or less, whereas 45 paths were completed within 10 to 20 μsec after the initial path request by the underlying processor. It was assumed that the path selection logic would always take 10 μsec to select a path, regardless of the length. Thus, no path can be acquired before the complete path has been specified by the path selection logic. So, for each of the 45 paths completed within 10 to 20 μsec, the first 10 μsec was assumed spent in selecting the path; the remaining time was utilized in acquiring the path one link at a time. Partially completed paths are not included in the table.

Table 6-6. Path Establish Time

TIME MICSEC	FREQ-UENCY
10	0
20	45
30	20
40	50
50	10
60	10
70	0
80	0
90	5
100	0
110	0
120	0
130	0
140	5

TOTAL SAMPLES 145 EXP. X VALUE 34.4 STD. DEV. 24.5

Table 6-7 lists the path length. The type number is the number of links in the path, e.g., a type 3 path contains three links. Here again, partial paths are not included in this table. This table brings to light a very desirable feature of this network. The node controllers are programmed to use the shortest paths. Two-thirds of the paths involve only one or two links.

Table 6-7. Path Length Frequency

TYPE	FREQ-UENCY
1	25
2	15
3	5
4	5
5	5
6	5

TOTAL SAMPLES 60 EXP. X VALUE 2.4 STD. DEV. 1.7

The final listing gives the transmission type (Table 6-8), which is a tabulation of the frequency of use for each type of message in the system. The type number corresponds to the following messages:

Type	Message
1	STX
2	LINKES
3	LNKBSY
4	GTLNK
5	BFRLNK
6	PRCRDY
7	GETPRC
8	(UNUSED)
9	PRCBSY

Table 6-8. Transmit Type Frequency

TYPE	FREQ-UENCY
1	60
2	860
3	35
4	895
5	520
6	60
7	145
8	0
9	85

TOTAL SAMPLES 2660 EXP. X VALUE 3.8 STD. DEV. 1.7

Some of these messages are used by the simulation program to signal events to the executive routine. For example, a STX message is generated whenever an underlying processor has finished processing a data block and needs to send the new data to another processor.

The most widely used message is the "GTLNK" (type 4) since this command is used to acquire the links for a path. Notice that a busy link is encountered less than 5 percent of the time (only 35 LNKBSY responses, type 3); however, over half of the time the requested processor (requested by a GETPRC command, type 7) is not available (85 PRCBSY responses, type 9). This is mainly due to bottlenecks at the ensemble averagers. BFRLNK commands proliferate since they are used to free the links when a transmission is completed and to terminate unsuccessful attempts to acquire paths.

Conclusions

The overhead is defined as the difference between the actual throughput time and the sum of the throughputs for each required process:

$$\text{Overhead} = T_{actual} - T_{theory}$$

where

$T_{theory} = T_{filter} + T_{FFT} + T_{EA} + 3T_{comm}$

T_{actual} = simulated actual delay

T_{filter} = filter delay

T_{FFT} = FFT delay

T_{EA} = ensemble averager delay

T_{comm} = time required to transmit a data block between processors.

The amount of overhead incurred by a data block is a function of the network topology and the number of each type of processor in the system. For the system as shown, overhead ranged from 10 to 100 percent of the theoretical throughput. The average was about 30 percent. A big bottleneck in this network is the ensemble averager. If these units were implemented with more input ports, better throughput would be achieved.

No attempt was made to optimize the processing flow. The optimum topology for this system would minimize the number of links between adjacent processing stages. In the network of Figure 6-18, the probability of completing a path would increase and the time needed to build a path would decrease. These improvements should reduce the overhead to around 6 percent of the theoretical delay, neglecting bottlenecking situations.

The "optimum" topology for a signal processing system is difficult to define due to the fact that each system emphasizes different features. The topology in Figure 6-18 minimizes throughput time but is still subject to bottlenecks at the ensemble averagers.

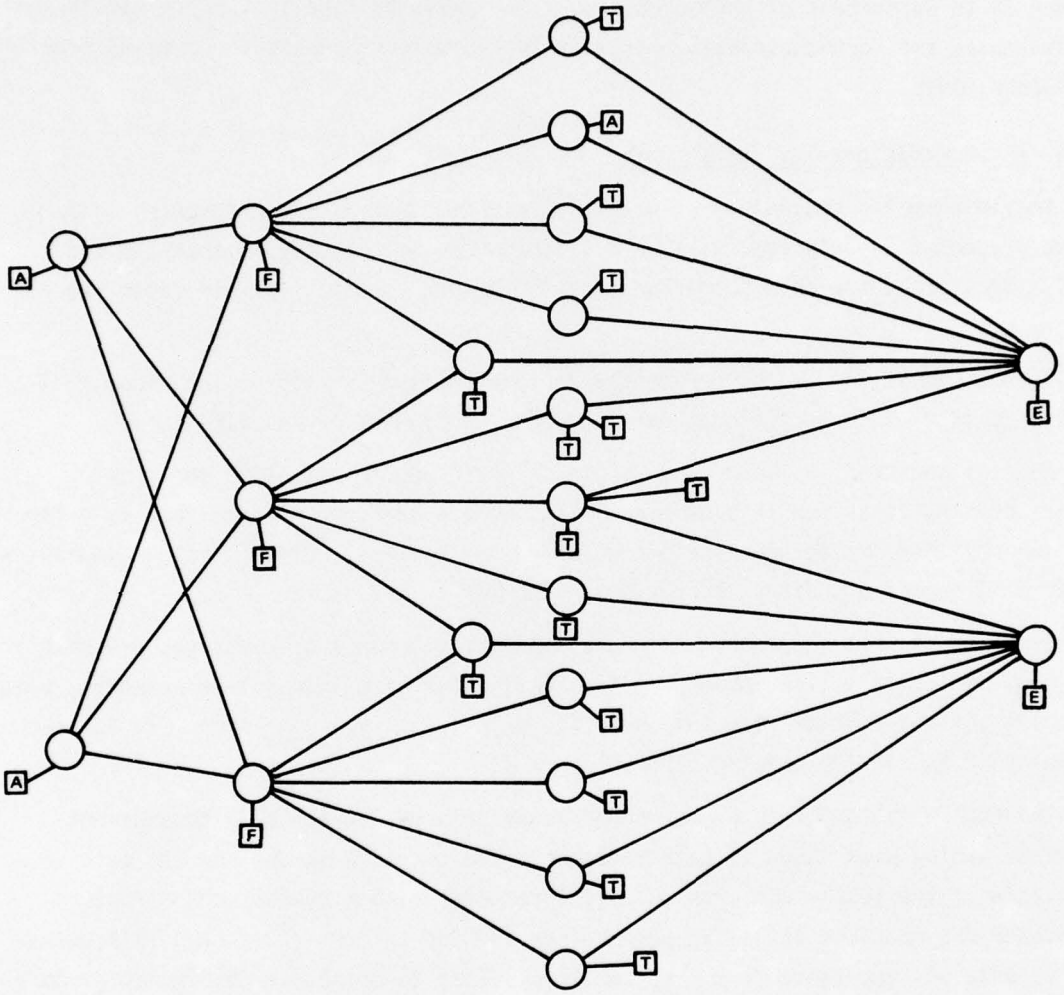


Figure 6-18. Improved SONAR Network

An encouraging result from the simulation was the fact that the node controllers were idle much of the time owing to their high speed and to the lack of more data-producing elements in the system. The nodes with highest utilization percentages were trying to create paths to the ensemble averagers which were already tied up processing previous data. In fact, all the data blocks go through the system up to the ensemble averagers with an average overhead of 6 percent of the throughput time up to that point. The addition of just two more ensemble averagers would have resulted in a total overhead of less than 8 percent of the theoretical time.

Much of the node controller's overhead (over 50 percent) is caused by execution of the path selection algorithm. In the detailed design of the node controller, much of the design effort was directed towards simplifying and increasing the speed of the hardware used in the final implementation. Actual path creation, which involves generating the appropriate GTLNK commands and interpreting the replies, accounts for another 35 to 40 percent of node overhead. The remaining time is used to update port status tables and scratchpad path matrix or to generate responses to commands received from other nodes.

6.3.2 Recommendations for Future Work

Determining the optimality of a complex system, such as the interconnection scheme presented in this report, is a difficult problem. Verifying that the end results are acceptable does not prove that the method used to generate those results is correct.

Even a simulation, such as reported in this section, serves only to show that the system is free from deadlocks for the configuration which was simulated.

Several additional simulations should be performed to establish the optimum network configuration for this problem. The network can then be perturbed to determine the relationship between the overhead and the network configuration. Simulations should also be performed for other types of networks.

Priority was not incorporated in the simulation because all messages presumably have equal priority in the SONAR problem. Definition of multifunction networks, with a significant grouping of data into separate priority classes, is necessary for such a simulation to provide meaningful results.

The other problem of interest is the behavior upon startup of a transparent interface system with adaptive path matrices. The question is whether the path matrices will eventually converge to full information about the network structure, or whether the matrices stabilize before learning the entire structure. This question is only of importance if the system is expected to change in configuration on a rapid basis; otherwise, fixed tables (i.e., PROM's) can be used without difficulty.

The results to date have been very encouraging; as indicated, several logical steps are to be taken next. These include additional simulations of increasing complexity to fully evaluate the potential of the transparent interface system.

6.4 DETAILED DESIGN RESULTS

Previous sections have described the algorithms which are to be implemented at each network node. In this section, two chip designs are presented: the crosspoint switch and path selection logic. The former is a 4 x 8 matrix of switches with integral control logic to simplify the implementation of any circuit switching network. The first approach to realizing the path selection logic was based on the Intel 8080 microcomputer; because it was too slow a dedicated logic implementation was performed as described in this section. It is amenable to realization on a single LSI circuit.

When the two circuits described here are developed the implementation of circuit switching networks for signal processor interconnection will be greatly simplified.

6.4.1 Crosspoint Switch

An important aspect of the system interface study performed on this contract is the identification of hardware functions required for efficient implementation of the connection algorithms. Although several functions have been identified the most critical is the crosspoint switch.

Crosspoint switches are required at each system node which has more than a single data link. This section describes the design and use of the crosspoint switch.

The basic crosspoint switch, as shown in Figure 6-19, consists of an 8 x 4 matrix of individual bilateral switches with the necessary logic to activate or deactivate the switches. Logic is included to permit connecting or disconnecting any pair of the eight ports and to connect any of the ports to any of the four trunks. The latter feature is used for expansion, as described later in this section. Although this design is intended to satisfy the path switching requirement of the transparent interface system, it is expected that this device will also satisfy many other requirments for circuit switching networks where either analog or digital communication is to be implemented.

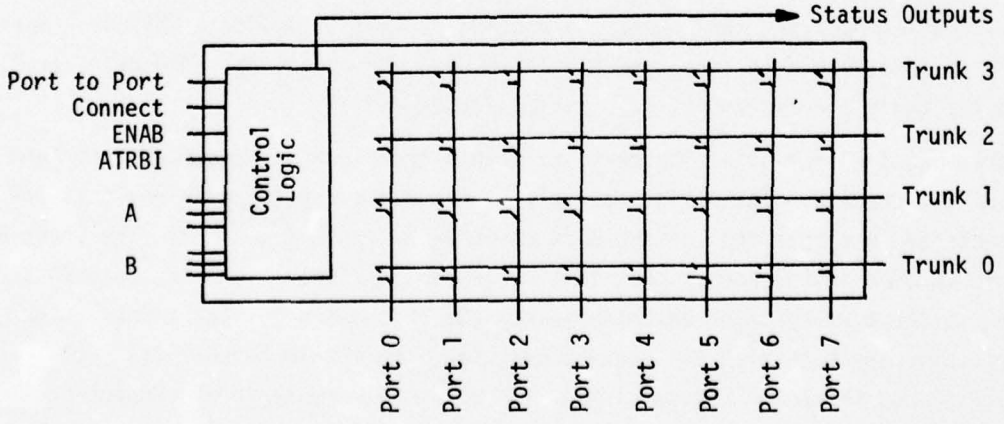


Figure 6-19. Basic Crosspoint Switch

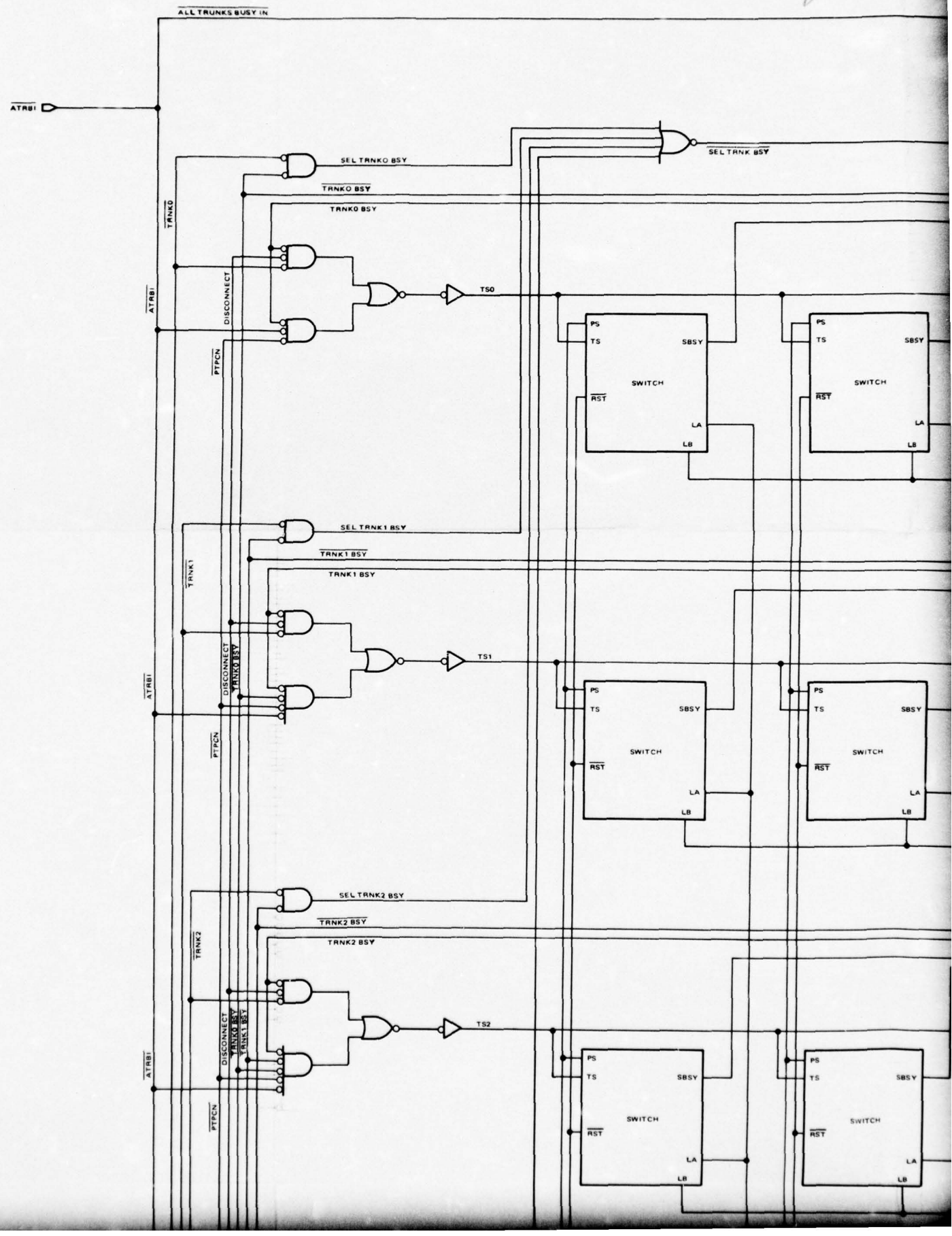
Detailed Design of the Crosspoint Switch

The crosspoint switch consists of an 8 x 4 array of field effect transistors which are controlled to connect selected pairs of the eight data ports. The switch implemented is a four-trunk, nonhierarchical, eight member switch of concurrency four. For eight port serial nodes a single chip is used; larger nodes (i.e., with more than eight ports) are implemented by cascading crosspoint switches both horizontally and vertically. A small amount of additional circuitry is required for cascading.

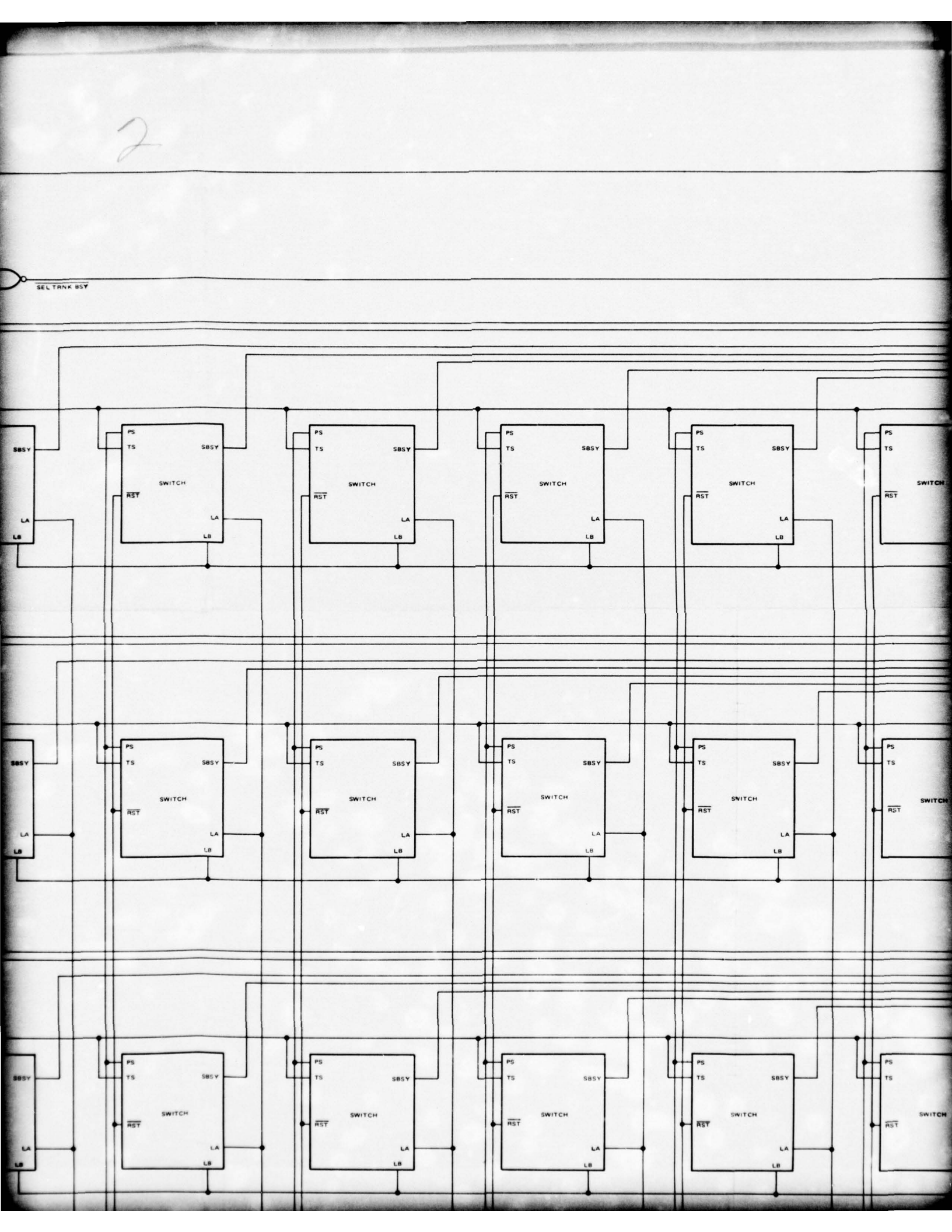
There are two types of connections possible with this chip: port-to-port and port-to-trunk. When performing a port-to-port connection, the two port numbers (ranging from 0 to 7) are presented to the crosspoint switch simultaneously, and the trunk used for the connection will be selected automatically. During a port-to-trunk connection, the port number and trunk number (ranging from 0 to 3) are presented to the chip, thus specifying which switch in the crosspoint is to be closed. This operation is performed when a connection between two horizontally cascaded chips is desired. The terms horizontal and vertical refer to the port lines and the trunks respectively. Thus, an 8 x 4 crosspoint has eight port lines and four trunks. If it were cascaded in the horizontal direction using another chip of the same configuration, it would have 16 lines and four trunks (16 x 4). If the port or trunk specified in the port-to-trunk operation is already in use, then the busy line is raised to notify the requester that the desired connection cannot be made. Similarly, in port-to-port operation, if either port is busy or if all the trunks are presently in use, then the busy line will be raised, indicating a busy condition.

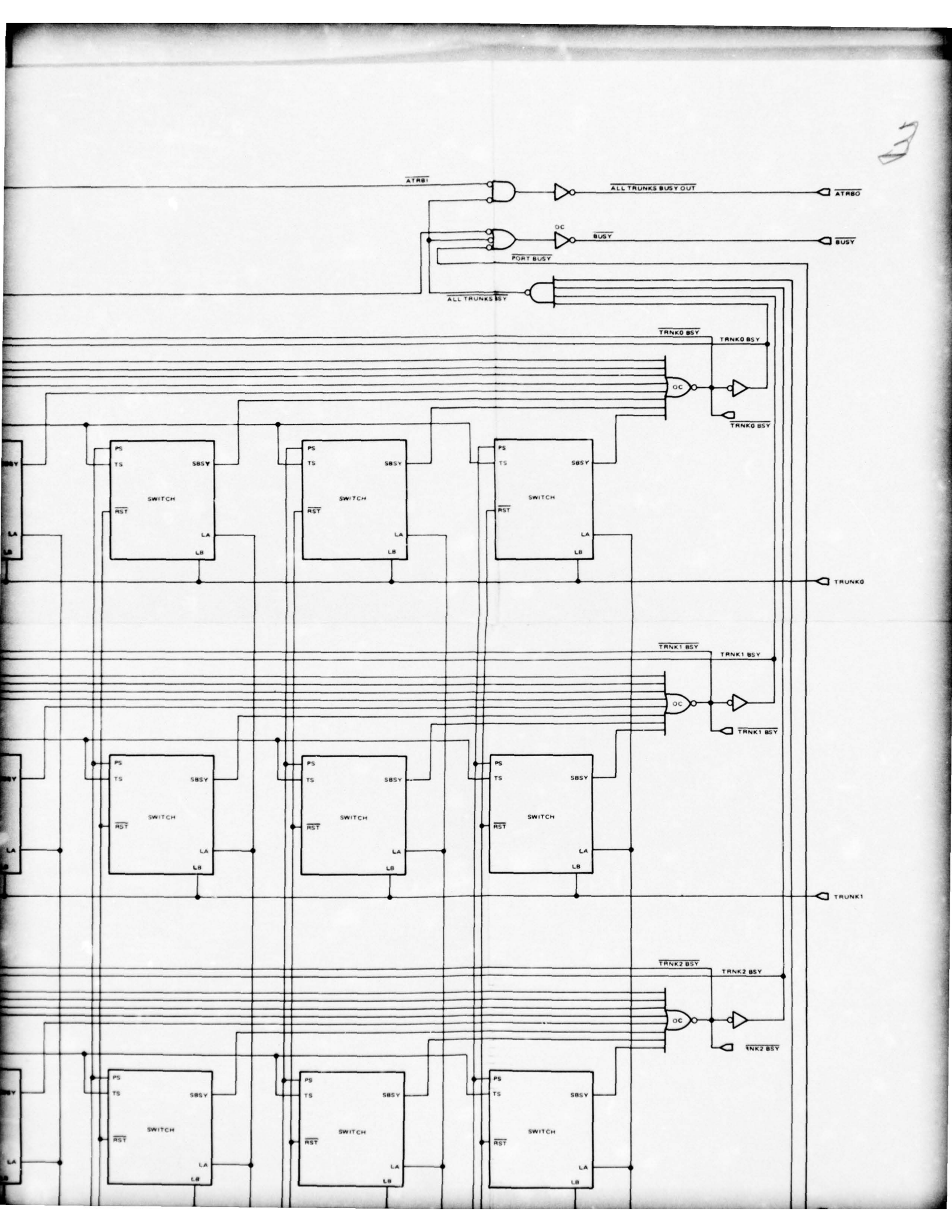
There are 28 functional pins on the crosspoint switch: four trunk lines labeled TRUNK0-TRUNK3; eight port lines (so named since they originate from the node ports) named PORT0-PORT7; three port A select lines (A0, A1, and A2); three port B or trunk select lines (B0, B1, and B2); an $\overline{\text{ENAB}}$ line used to enable the decoders for the select lines; a PORT-TO-PORT line which is high for a port-to-port connection and low for a port-to-trunk connection; the $\overline{\text{CONNECT}}$ line which is used to indicate whether a make or break is desired; the busy line; and six lines used for cascading. The detailed logic diagram for the 8 x 4 crosspoint is shown in Figure 6-20.

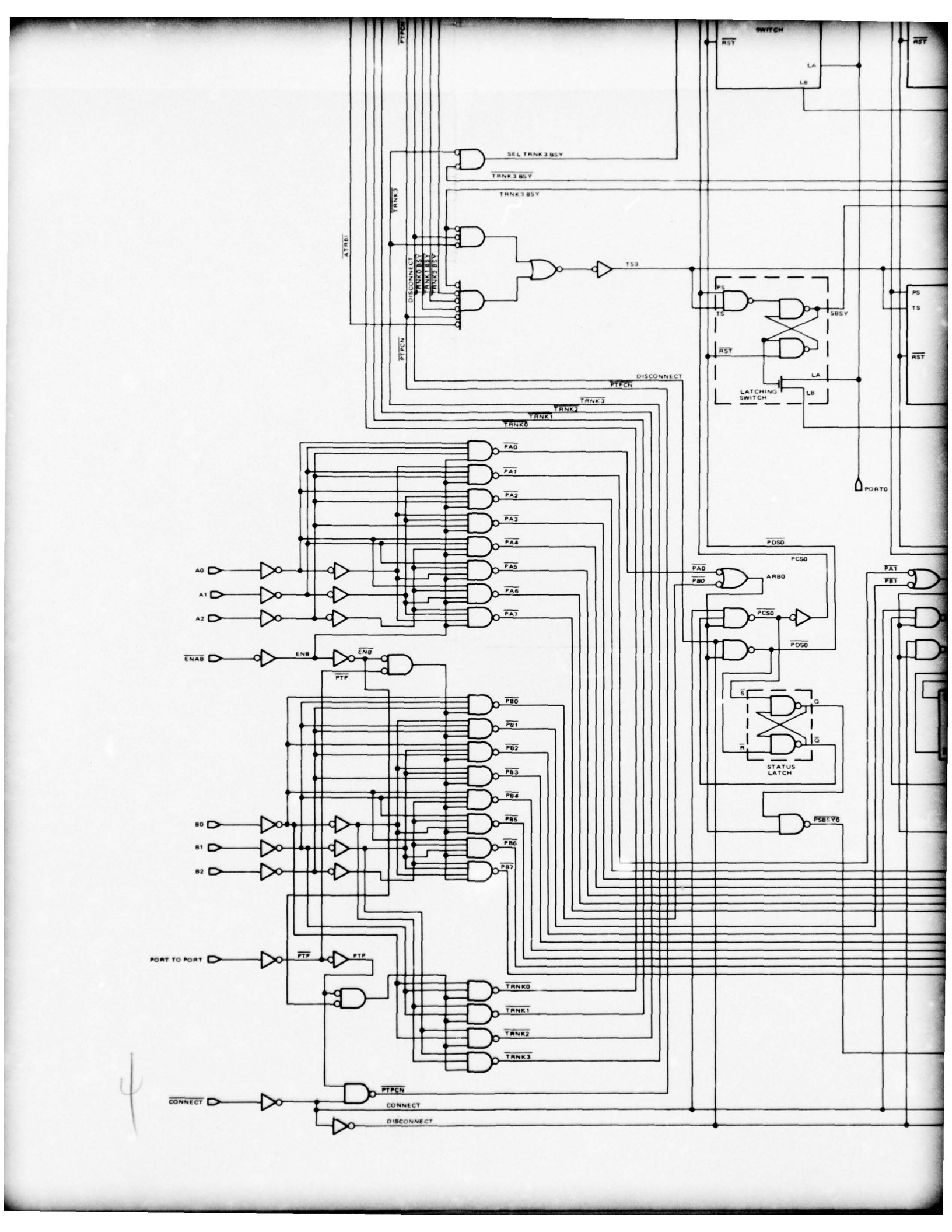
Three decoders for selecting ports or trunks are shown in the lower left-hand corner of the drawing. The A lines are always decoded since there is at least one port specified in either the port-to-port or port-to-trunk connection. The three B lines are decoded as the second port in the port-to-port operation, but only B0 and B1 are used in the port-to-trunk operation since the chip has only four trunks. Thus, the upper decoder is enabled for every connection, the middle decoder for port-to-port connections, and the lower decoder is enabled during a port-to-trunk connection.

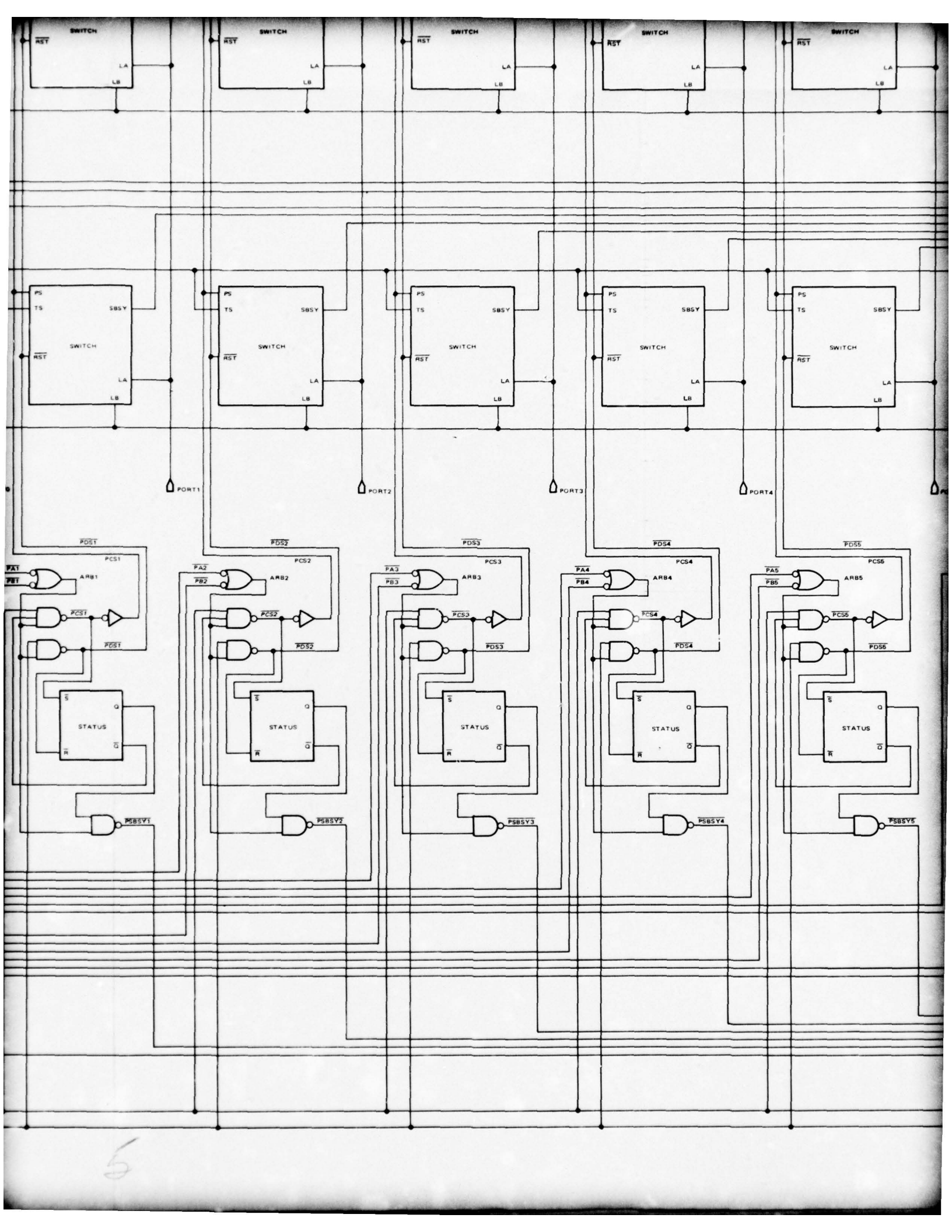

ALL TRUNKS BUSY IN
ATRBI
SEL TRNKO BSY
TRNKO BSY
TRNKO BSY
TRNKO
ATRBI
DISCONNECT
PTPCN
TS0
SEL TRNK BSY
PS
TS
SBSY
SWITCH
RST
LA
LB
SEL TRNK1 BSY
TRNK1 BSY
TRNK1 BSY
TRNK1
TS1
SEL TRNK2 BSY
TRNK2 BSY
TRNK2 BSY
TRNK2
TS2

2




ATRBI
ALL TRUNKS BUSY OUT
ATRBO
OC
BUSY
BUSY
PORT BUSY
ALL TRUNKS BSY
TRNK0 BSY
TRNK0 BSY
OC
TRNK0 BSY
PS
TS
SBSY
SWITCH
RST
LA
LB
TRUNK0
TRNK1 BSY
TRNK1 BSY
OC
TRNK1 BSY
TRUNK1
TRNK2 BSY
TRNK2 BSY
OC
TRNK2 BSY


SWITCH
RST
LA
LB
SEL TRNK3 BSY
TRNK3 BSY
TRNK3
ATRB1
DISCONNECT
TRNK0 BSY
TRNK1 BSY
TRNK2 BSY
TRNK3 BSY
TS3
PS
TS
SBSY
LATCHING SWITCH
PTPCN
DISCONNECT
TRNK3
TRNK2
TRNK1
TRNK0
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PORT0
PDS0
PCS0
ARB0
A0
A1
A2
ENAB
ENB
PTP
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
STATUS LATCH
B0
B1
B2
PORT TO PORT
CONNECT
DISCONNECT


SWITCH
RST
LA
LB
PS
TS
SBSY
PORT1
PORT2
PORT3
PORT4
PDS1
PCS1
PA1
PB1
AR81
PCS1
PDS1
STATUS
S
R
Q
PSBSY1
PDS2
PCS2
PA2
PB2
AR82
PCS2
PDS2
PSBSY2
PDS3
PCS3
PA3
PB3
AR83
PCS3
PDS3
PSBSY3
PDS4
PCS4
PA4
PB4
AR84
PCS4
PDS4
PSBSY4
PDS5
PCS5
PA5
PB5
AR85
PCS5
PDS5
PSBSY5

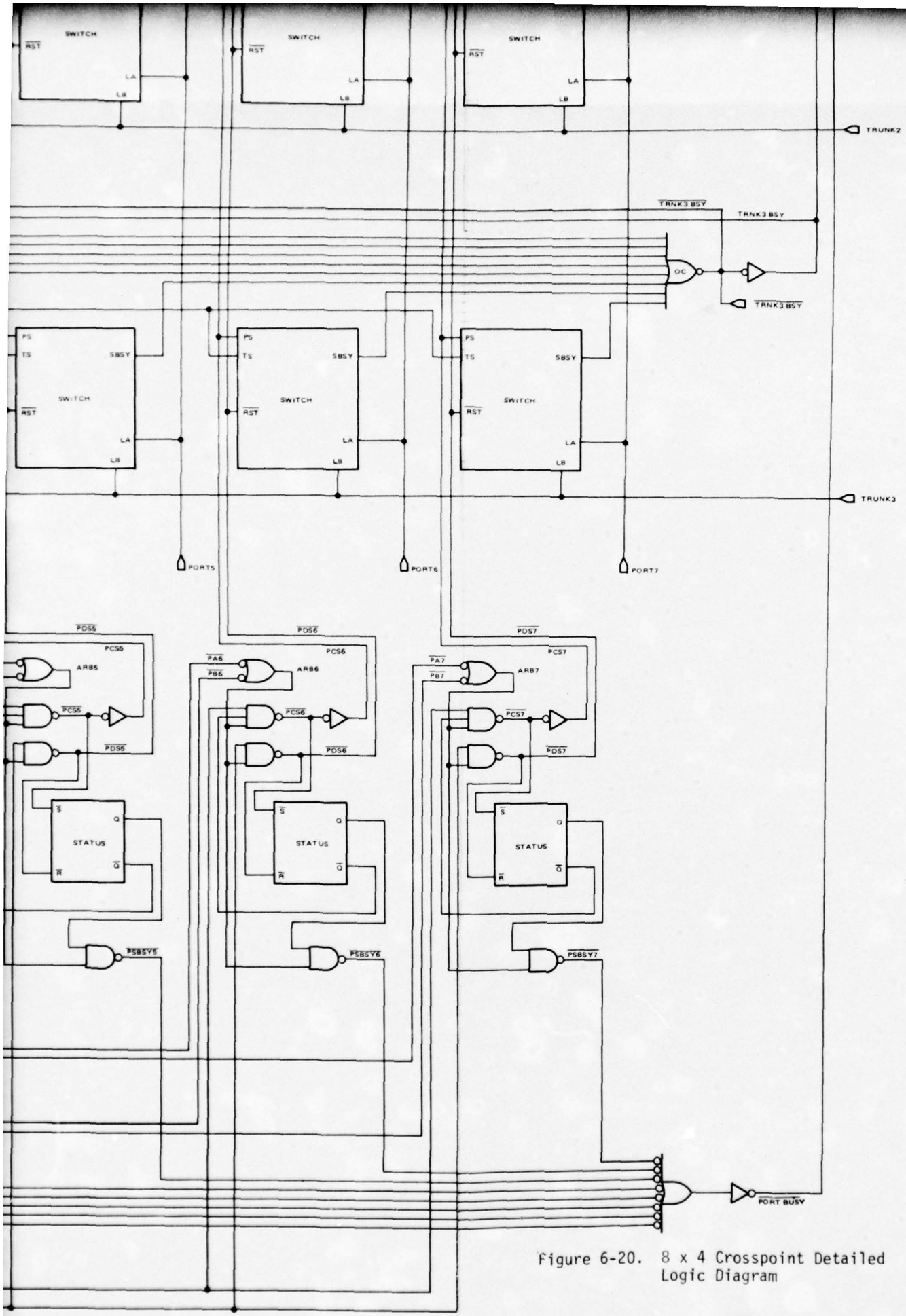


Figure 6-20. 8 x 4 Crosspoint Detailed Logic Diagram

The latching switch is used to make a connection between a port line and a trunk but it also contains a switch busy (SBSY) line. The switch is turned on by strobing PS and TS simultaneously, which raises SBSY and turns the switches on, thus connecting the port line (LA) and the trunk (LB). The switch is turned off by lowering $\overline{RST}$, which resets all the switches in a vertical column. All the SBSY lines for each trunk are NORed together to produce a $\overline{TRNKi\ BSY}$ line which is used during horizontal cascading.

The system controller determines if a port line is busy by referring to the eight status latches located along the bottom of Figure 6-20. Each status latch is simply an $\overline{R}$ - $\overline{S}$ latch which is set if this port is selected (ARBi=1) for a connect operation (CONNECT=1) and if the latch is not already set ($\overline{Q}$=1). The latch is reset if this port is selected (ARBi=1) for a disconnect operation (DISCONNECT=1). (Note that the disconnect operation resets all four switches associated with this port.) If it is a connect operation and the port is already in use (Q=1), then $\overline{PSBSYi}$ is forced low which in turn forces $\overline{PORT\ BUSY}$ low.

When a status latch is set, a pulse is sent along the line which connects all four PS inputs together for that port. However, a similar pulse is needed on the TS line for one of the switches in order to make a port-to-trunk connection. The set of gates at the left end of each trunk line is used to generate a TS pulse. For a port-to-trunk operation, the middle ANDN gate must be used. This gate has inputs from three different sources: the DISCONNECT line, TRNKi BSY line, and $\overline{TRNKi}$ line. The port-to-port operation does not specify which trunk is to be used for the connections. Therefore, the chip must make this decision. The trunk selection logic picks the lowest numbered, unused trunk. This logic is implemented with the lower ANDN gate of the four-gate cluster located at the left end of each trunk. Using trunk two as an example, we see that the inputs to the gate in question are TRNK2 BSY, $\overline{TRNK1\ BSY}$, $\overline{PTPCN}$ and $\overline{ATRBI}$. This trunk will be selected for a port-to-port operation if:

- It is not currently being used (TRNK2 BSY=0)
- All preceding trunks (one and zero) are already in use ($\overline{TRNK1\ BSY}$ and $\overline{TRNK0\ BSY}$=0)
- The operation being performed is a port-to-port operation ($\overline{PTPCN}$=0)
- The chips are vertically cascaded, and all the trunks in the chips below this particular chip are already in use ($\overline{ATRBI}$=0).

As can be seen from Figure 6-20, trunk zero will always be chosen first for a port-to-port connection, then trunk one and so on.

In the event that one or both of the ports in the port-to-port operations, or the port or trunk in the port-to-trunk operation are busy, then the busy flag is raised. The three signals that contribute to the generation of $\overline{BUSY}$ are: $\overline{SEL\ TRNK\ BSY}$, which

PRECEDING PAGE BLANK-NOT FILMED

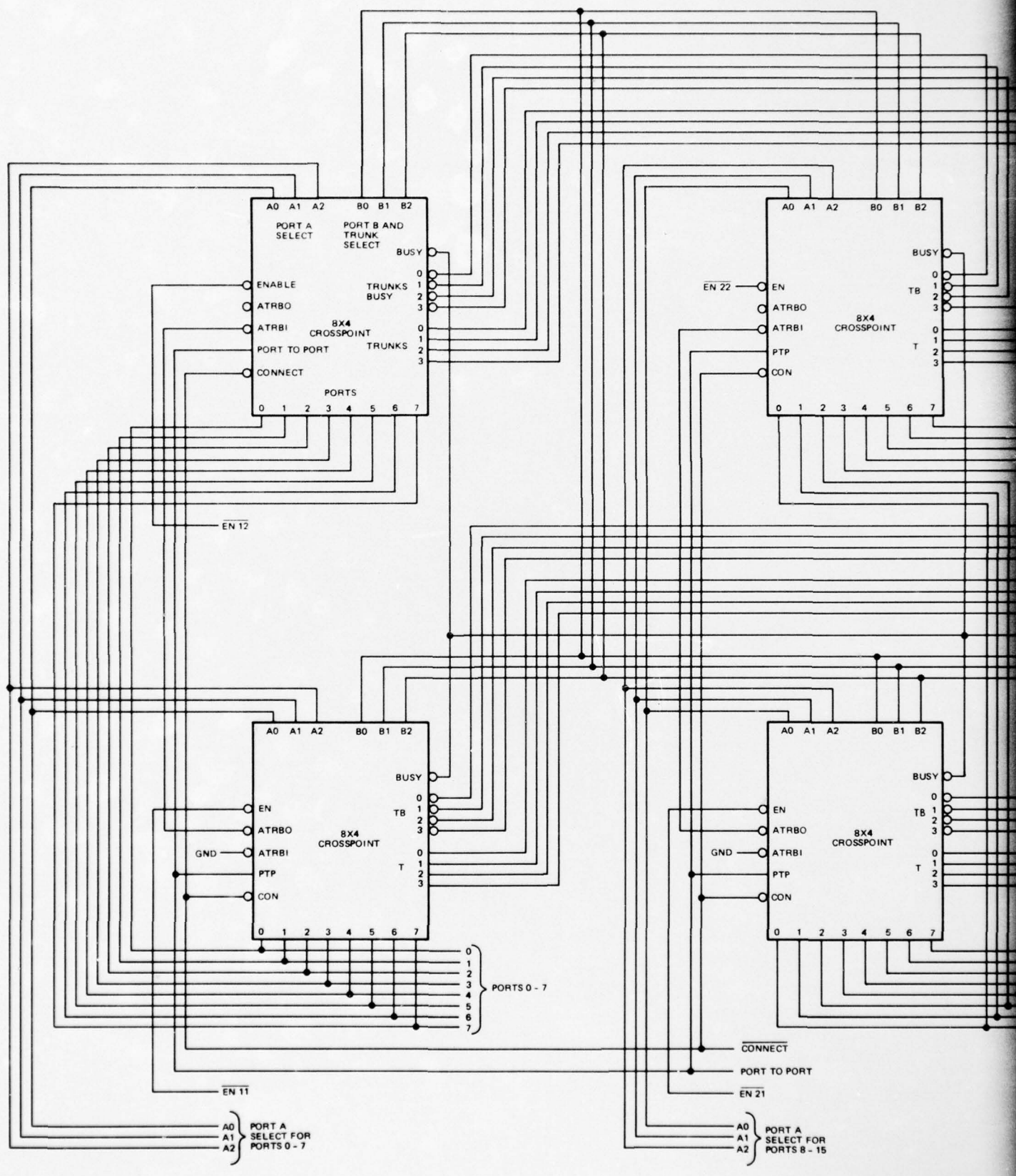
goes low if the trunk specified in the port-to-trunk operation is busy; $\overline{\text{ALL TRNKS BSY}}$, which goes low during a port-to-port operation if all trunks in every chip preceding this one (in this column) are being used ($\overline{\text{ATRBI}}=1$) and all four trunks in this chip are in use; and $\overline{\text{PORT BUSY}}$, which goes low if any port specified in either type of connection is presently occupied.

Expansion

A 16 x 8 crosspoint switch is shown in Figure 6-21. Notice that all the chips in a column have their port A select lines tied together and that the port B select lines for all chips are tied together. With these connections extra gates are needed to ensure that the proper switches are enabled. In port-to-port operation, all the switches in the column corresponding to the specified ports are enabled (e.g., when connecting port 9 to port 13, chips 21 and 22 must be enabled). For port-to-trunk operation, all the chips in the row corresponding to the specified trunk are enabled (e.g., if trunk 5 is specified, then chips 12 and 22 are enabled). It is also necessary to connect the trunk busy lines for each row to prevent two different chips from trying to use the same trunk for two different port-to-port operations.

When connecting two ports together that do not belong to the same chip (e.g., port 3 to port 11) two port-to-trunk operations must be performed, one for each port. However, both connections must be made at the same time, otherwise the initial port-to-trunk operation will lock out any future attempts to connect to the specified trunk. With the present chip, a race condition occurs when performing simultaneous port-to-trunk operations. There are ten gate delays between the initiation of a port-to-trunk operation and the falling of $\overline{\text{TRNKi BSY}}$ plus one more gate delay to raise TRNKi BSY on the other cascaded chips. If TRNKi BSY is raised before a TSi pulse is generated, then only one port-to-trunk operation will succeed, locking out the other desired connection. To prevent this, the inverter connected to $\overline{\text{TRNKi BSY}}$ on the right-hand side of Figure 6-20 could be replaced with the circuit of Figure 6-22. The signals $\overline{\text{LTRNKi BSY}}$ and LTRNKi BSY would replace the present $\overline{\text{TRNKi BSY}}$, and TRNKi BSY which are used in the trunk selection logic at the left-hand side of Figure 6-20. The D flip-flop used as the busy latch should be rising-edge triggered so that the value of $\overline{\text{TRNKi BSY}}$ before the simultaneous port-to-trunk operation is used.

A minor problem exists with the cascaded configuration of Figure 6-21 concerning $\overline{\text{BUSY}}$. Assume that a port-to-port connection is desired for port 1 and 5 and that the only trunk available is trunk 6 (which is in chip 12). When chips 11 and 12 are enabled, chip 11 discovers that all its trunks are busy and lowers $\overline{\text{ATRBO}}$, thus signaling chip 12 to try one of its trunks. Chip 12 will use trunk 6 since it is the only trunk available. However, $\overline{\text{BUSY}}$ will have gone low because all of the trunks in chip 11 are busy. This would indicate that the connection had not been made which


A0 A1 A2 B0 B1 B2
PORT A SELECT
PORT B AND TRUNK SELECT
BUSY
ENABLE
TRUNKS BUSY
ATRBO
ATRBI
8X4 CROSSPOINT
PORT TO PORT
TRUNKS
CONNECT
PORTS
0 1 2 3 4 5 6 7
EN 12
EN 22
EN
TB
PTP
CON
T
GND
EN 11
EN 21
CONNECT
PORT TO PORT
PORTS 0 - 7
A0
A1
A2
PORT A SELECT FOR PORTS 0 - 7
PORT A SELECT FOR PORTS 8 - 15

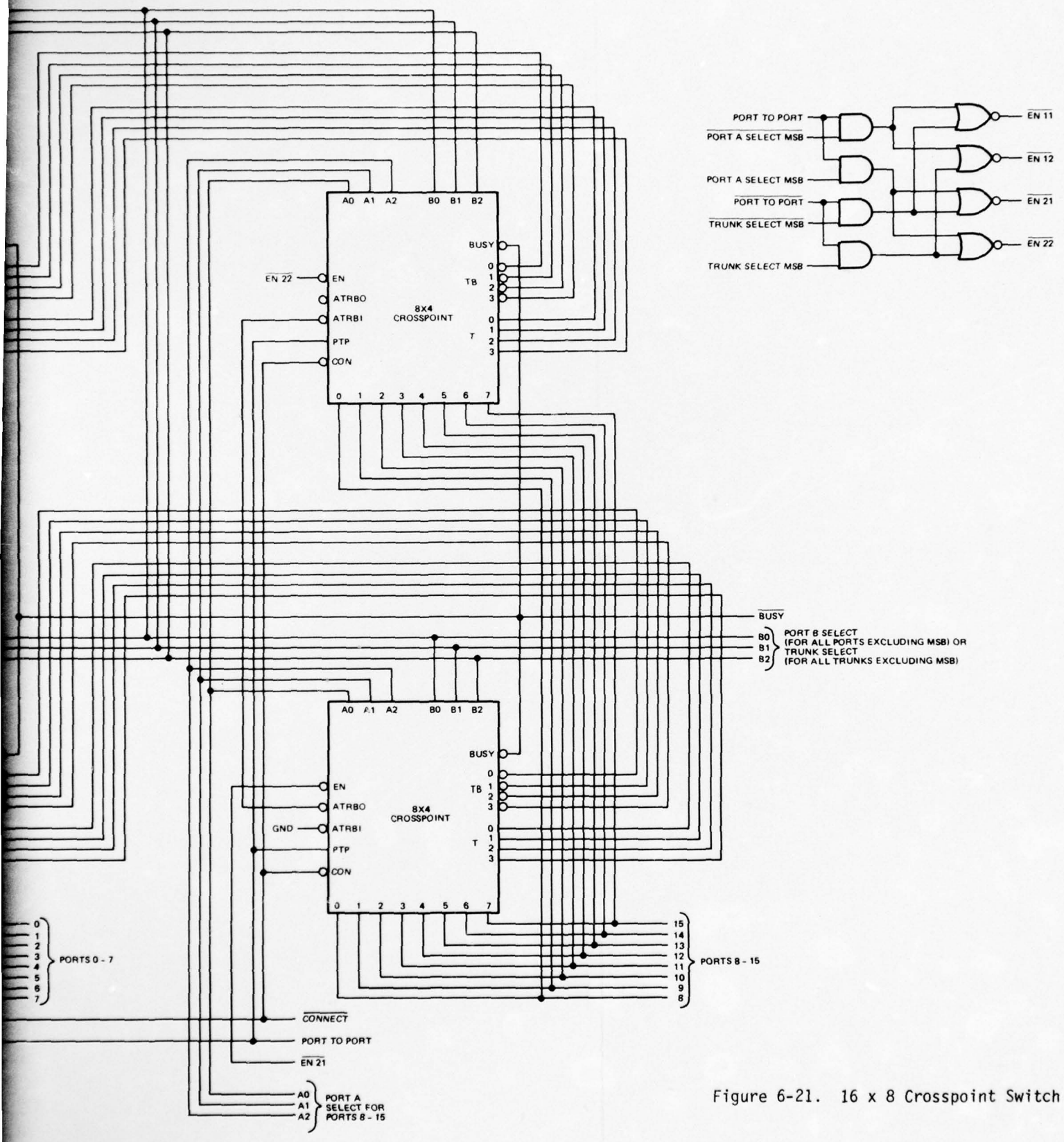


Figure 6-21. 16 x 8 Crosspoint Switch

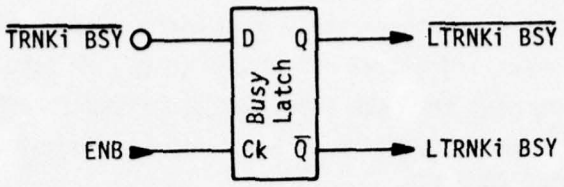


Figure 6-22. Latch for Expansion of Trunk Busy Signal

is, of course, in error. One solution to this problem is to add the external circuit which is diagrammed in Figure 6-23. These three gates are all that is needed for any number of switches cascaded in the same manner as in Figure 6-21.

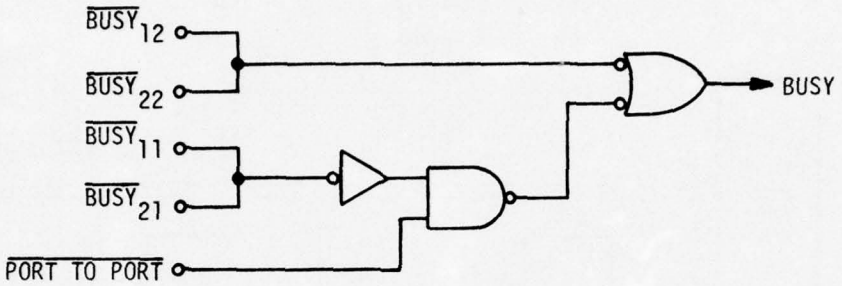


Figure 6-23. Circuit for Expansion of Busy Signal

Recommendations for Future Work

In the definition and detailed design of this chip two minor problem areas have been discovered as described in the preceding section. Although solutions have been described, it is felt that further design effort can yield "cleaner" solutions. A complete logic simulation should also be performed to verify the design and to generate test sequences.

6.4.2 Path Selection Logic

The path selection algorithm described in Section 6.2 can be implemented with either a microprocessor (e.g., an Intel 8080) or with dedicated logic. In the initial phase of the study an Intel 8080 system was investigated and is summarized here. Although it established feasibility for the algorithms, its delay (on the order of 300 μsec to select a path) was deemed excessive.

A dedicated hardware approach was investigated and is described in detail in this Section. It comprises two sections: path selection logic and path selection controller. First order timing indicates that this approach can select a path in under 10 μsec.

Microprocessor Implementation

A node consists of a central node controller and some number of ports. At each port a line interface accepts data from a communication channel (link) and presents it to the node controller. The line interface also takes data from the node controller and sends it over the link.

PRECEDING PAGE BLANK-NOT FILMED

The node controller consists entirely of a micro or special purpose processor that is conducive to stack and matrix operations (e.g., an Intel 8080) to perform the path matrix construction and the path selection algorithm (see Figure 6-24). However, there will also be a moderate quantity of ancillary hardware for any processor such as clock generators, RAM's, ROM, etc.

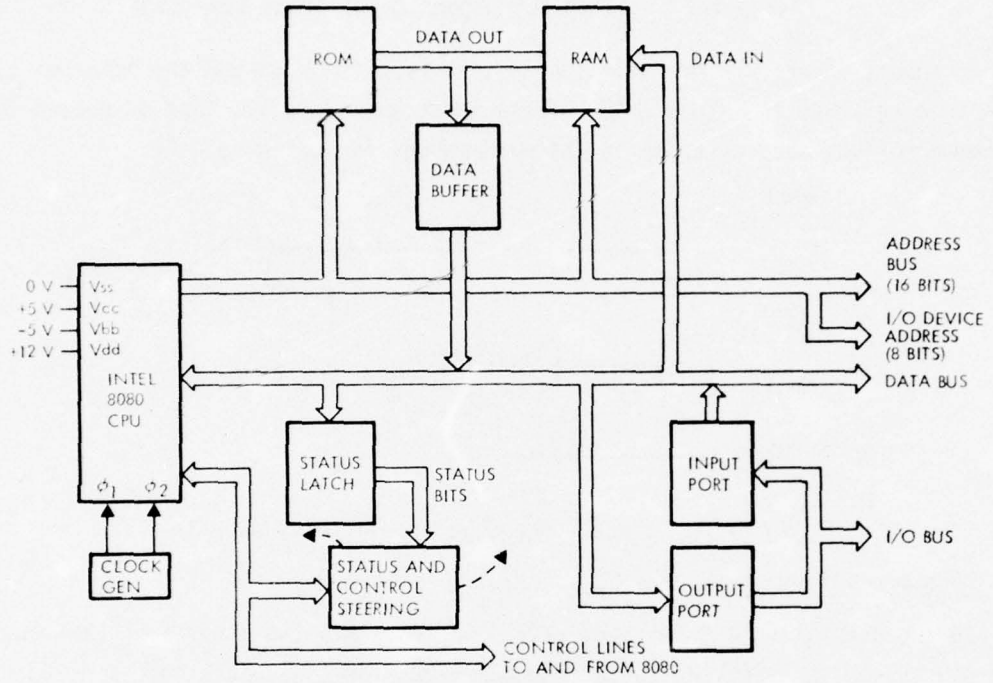


Figure 6-24. Node Controller

The line interface units (Figure 6-25) consist of a line buffer for the communication channel, a UAR/T for receiving and transmitting the data, two FIFO's (one for input data and the other for output data), a box for detecting special messages, and a box to control the interface functions.

The line buffer consists mostly of line drivers and receivers, but will also select the output data source (i.e., the UAR/T or the switch). The UAR/T performs the data detection and conversion from serial to parallel. It also accepts parallel output data, converts it to serial, and transmits the data with the proper start and stop bits. The two FIFO queues are used for temporarily storing multiword commands that are used by the nodes to establish paths.

Each node has the ability to connect any port to any other port. This crosspoint, nonhierarchical switch of concurrency M/2 (where M is the number of ports) is implemented using the crosspoint switch described in the previous section.

The control box decodes and executes the control commands from the processor and senses the interface status which is continually passed back to the node controller.

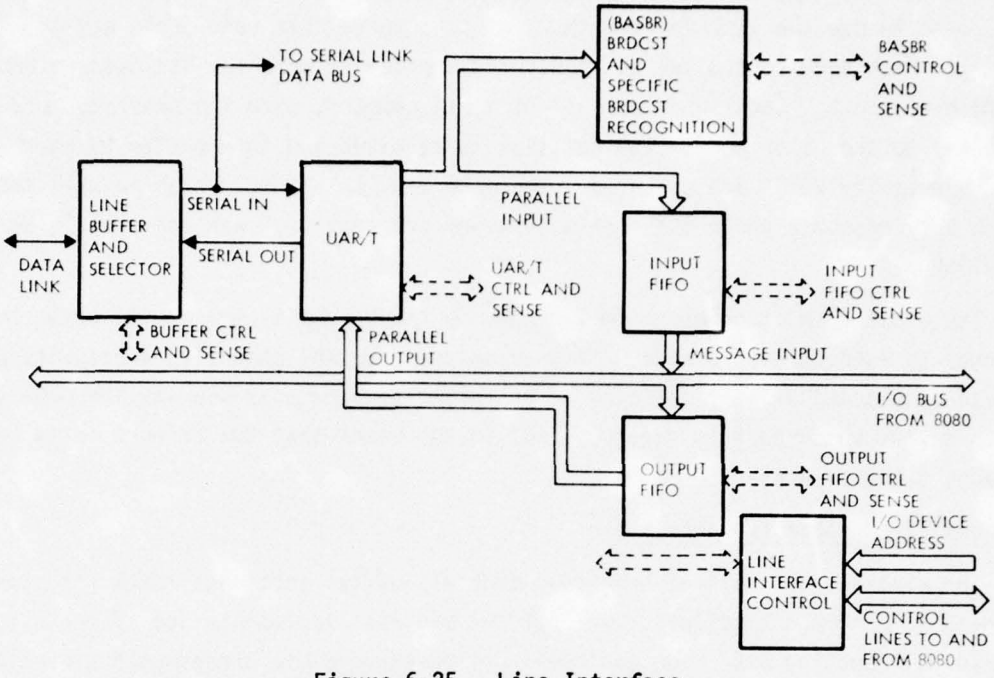


Figure 6-25. Line Interface

Two types of priority are used through this system: (1) dynamic priority, associated with a message sent by a node; and (2) static priority associated with each port in a node. When a node has a message to send to another member of the network, it generates a priority word for that message. This priority is dynamic in two ways: (1) if the links it wishes to use are in use, it can raise its priority until those links are obtained, or (2) if the message it is sending contains a section that is not very important, it can change the priority during the message with the option of raising the priority later in the message.

Two kinds of priority conflicts arise in this system. The first conflict occurs when a node controller is trying to create a path using a link that is presently employed by a different node. The busy link has a priority word stored in the memory of the node controller. This priority word is compared with the priority word of the interfering node. If the busy link has a higher priority, then a "link busy" response is sent to the interface. If on the other hand, the busy link is in use at lower priority, then an interrupt is sent to the link users informing them that the link will be preempted. After an amount of time determined by system protocol, the link is then assigned to the interferor and its priority word is set to the priority of the new originator.

The second kind of conflict occurs when two nodes, one on each end of a free link, want to use the link for a path. Let us suppose that both nodes act simultaneously. Each node controller has generated a priority word for its message and has determined a path. Each sends out a "get link" command, with the priority word attached, to the other node. The get link is received and interpreted by each node and the priority words are compared. The node with the higher priority will send a link busy response while the node with lower priority will respond with "link established."

The other form of priority implemented in the system is the static priority attached to each port of a node. This priority is established with a priority encoder and can be changed only by rewiring. This priority scheme is used by the node controller to decide which port to service first in the event that two or more ports receive messages simultaneously.

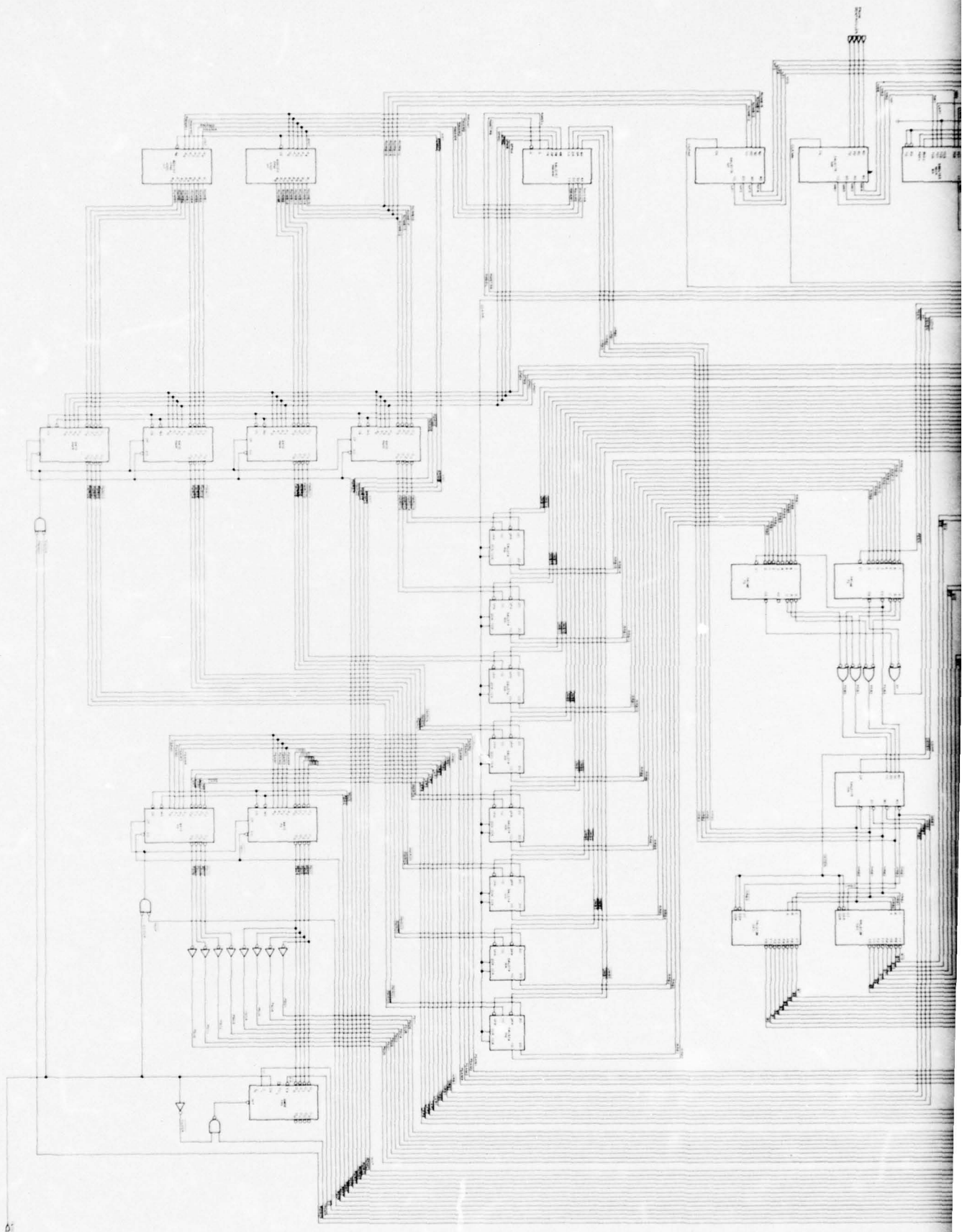
Dedicated Logic Implementation

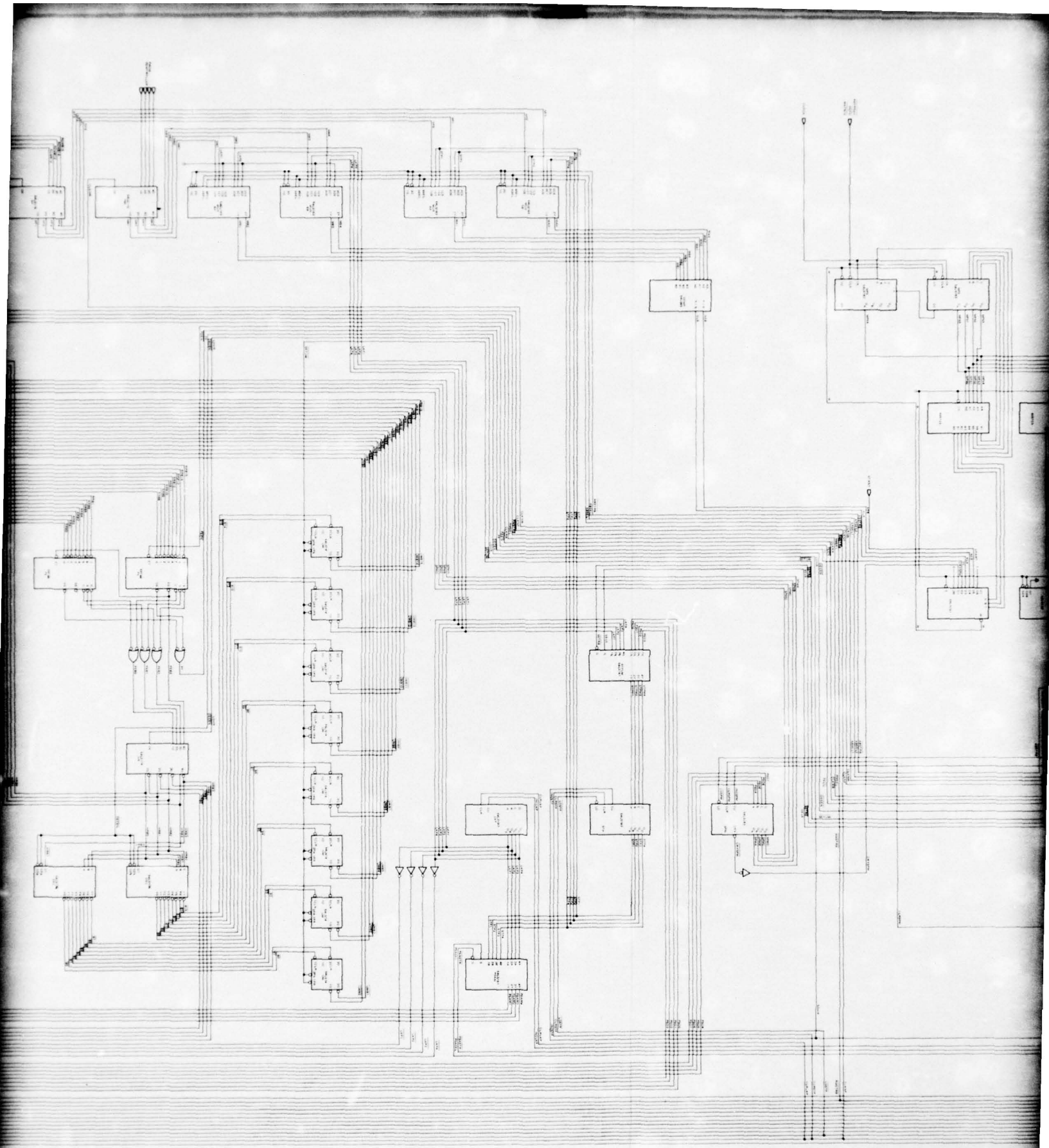
The special purpose implementation consists of two sections: path selection logic and path selection controller. The logic is a direct implementation of the algorithms described in Section 6.2. The controller is based on a microprogrammed sequential network* which is a powerful technique for implementing sequential control units.

Path Selection Logic. The logic described herein is an MSI implementation of the path creation algorithm. This particular design can be used with systems employing up to 16 nodes. However, the design is easily extended to larger networks.

The design centers around five main structures: permanent path matrix and name table ROM, scratchpad path matrix, node selection logic, path stack, and threaded path stack. The actions performed by these structures are coordinated by a microprogrammed controller (described in the next section of this report). The path selection hardware as shown in Figure 6-26 (see also the simplified version of this drawing shown on Figure 6-27), was designed for a static network. For this reason, the permanent path matrix (PPM) is contained in a ROM. Each time a message is to be transmitted, the scratchpad path matrix RAM (PMR) must be refreshed. On command from the central controller, the information contained in the lower half of the ROM is transferred to the PMR. The path matrix in the PMR describes the current topology of the network as seen

*E.E. Swartzlander, Jr., "Microprogrammed Sequential Networks," New Components and Subsystems for Digital Design, (Technology Service Corp., Santa Monica, Calif.) 1975, pp. 111-114.





Fi

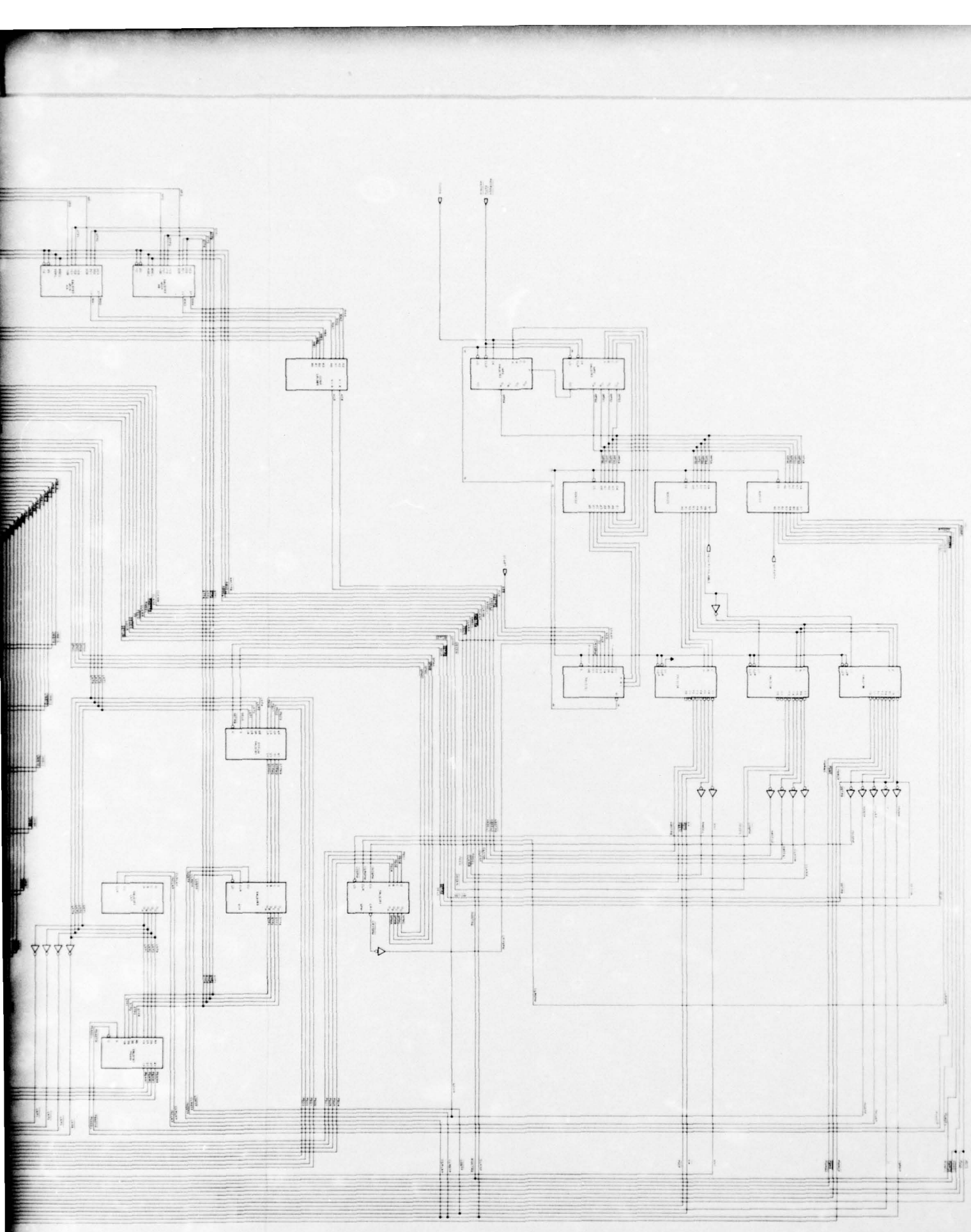


Figure 6-26. Detailed Path Selection Logic Diagram

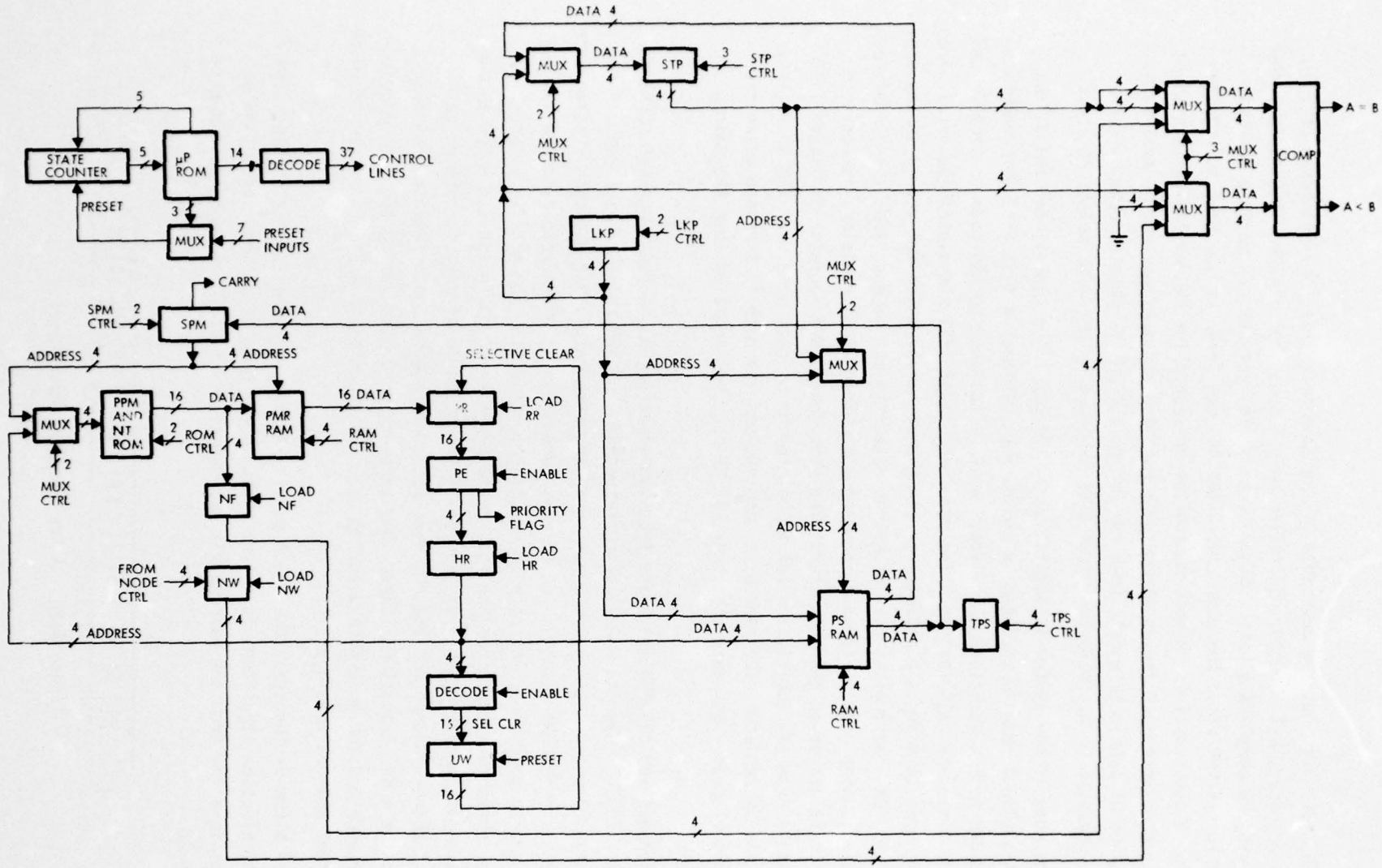


Figure 6-27. Simplified Block Diagram of Path Selection Logic

PRECEDING PAGE BLANK-NOT FILMED

from this node. It would be desirable if the path matrix was only composed of links that are not busy, but the node controller only learns that a given link is busy when it is trying to complete a path. Other links in the network may be busy but not recorded in the PMR since the node controller has not tried to use them during the current path construction. A node controller refreshes the PMR periodically because the links that were busy during former path building attempts may now be available. The transfer of path matrix data from the PPM to the PMR is usually done before a path is requested to minimize the total time consumed by the path selection logic.

Each node in the system is identified by two names: a global name and a local name. The global name is assigned by the network designer and is used for communication between node controllers. The local name is assigned by the node controller and is used during path calculations. The node name table (NT) translates the local names, which are used to address the table, to the global names, which are stored in the upper half of the ROM. Before the path selection algorithm can be executed, the global name of the destination node is strobed into the node wanted (NW) register. Whenever a local name is placed on the path stack, the node found (NF) register is loaded with the contents from the location in the NT pointed to by that name. If the NF register matches the NW register, the search is complete and the path is threaded using the link field of each stack word. However, if NF does not equal NW then the search continues.

Addresses for the PMR originate from two sources: PMR address register (SPM) and path stack (PS). When the PMR is being refreshed, the SPM register is first cleared and then incremented until every location in the path matrix RAM has been written into from the corresponding location in the PPM. When data is being read from the PMR during the search portion of the path selection algorithm, the SPM acts as a latch for the local names extracted from the PS. These names are used to address a row in the scratchpad path matrix which will provide the next source of interconnection data. One final address source for the PMR does not stem from the path selection hardware, but from the node controller. These addresses are generated during the path acquisition sequence and are used when altering the scratchpad path matrix to correspond with the current network topology.

When a row of the scratchpad path matrix is read from the PMR during execution of the path selection algorithm, it is presented to the node selection logic. The bit positions in a row from the path matrix are numbered from left to right as shown in Figure 6-28.

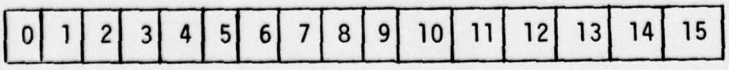


Figure 6-28. A Row from the Path Matrix

The bit positions correspond to the local names given each node by the node controller. A "1" denotes a connection between two nodes. For example, from Figure 6-29, there are eight links that lead away from node 1.

	node a	node b	node c	node d	node e	node f	node g	node h	node i	node j	node k	node l	node m	node n	node o	node p
node i	0	1	1	0	1	0	0	1	0	1	0	1	1	0	0	1

Figure 6-29. Node i Connectivity

Thus, during path selection, all eight links would be considered candidates for the complete path. However, to avoid circular paths, no link may be nominated more than once during the path search operation. Thus, the node selection logic performs two functions: it identifies the links that are being considered for a path, using the local name of the node at which the link terminates (from the searching node's point of view); and it prevents a connection from being used more than once for any path by marking a node "used" whenever it is nominated. The used word (UW) register contains a used bit for each node in the network which identifies the nodes being considered for a path. The bit corresponding to a candidate node is set to "0" when that node is nominated. All bits corresponding to nodes not under consideration remain at "1". The UW register is numbered the same as a row from the path matrix.

As a row is read from the PMR, it is placed in the row register (RR). All 16 used bits from the UW register are connected to the corresponding clear lines at the RR. If a node has already been included in the path stack, its bit in the RR will be forced to "0" regardless of its value in the PMR.

The outputs of the RR are fed to a priority encoder which determines the position of the left-most "1" and outputs the local name assigned to that position. The local name is then loaded into a holding register (HR) to avoid losing this information when the final section of the node selection logic is enabled. The last section consists of a 4-line-to-16-line decoder whose outputs are connected to the clear inputs of the UW. Thus, the four-bit local name energizes one of the decoder output lines which in turn sets the corresponding used bit to "0". The used bit clears the RR bit and the priority encoder determines the new left-most "1". This iteration continues until every "1" in the RR has been cleared. If the destination node was not among those selected from the present path matrix row, then a new row must be processed by the node selection logic.

The HR outputs are used for two other operations besides the node selection logic. First, the local name held in the HR is used to address the name table to yield the equivalent global name, which is loaded into the NF register, and then compared to the NW register. Second, the latched local name is positioned in the name field of the stack word.

The path stack is a RAM that is addressed by a pointer (the stack pointer, STP) which is incremented each time a word is to be written into memory. Each word in the stack consists of a name field and a link field (see Figure 6-30).

NAME FIELD	LINK FIELD

Figure 6-30. Stack Word

The name field is filled with the local name of a node while the link field contains a pointer back to a lower position in the stack. The node specified by the name field of the lower stack position was used as a "point-of-search" from which new candidate nodes, if any, could be discovered (by scanning the lower node's row in the path matrix). These new candidates would all be linked back to their "point-of-search" node using the stack position of the point-of-search node. For example (Figure 6-31), if the point-of-search node "E" finds two new candidates, "S" and "R", then when they are included in the path stack their link fields will be equal to E's position in the stack (which is the current value of the link pointer, i.e., 2). When a new point-of-search node is required, the link pointer (LKP) is incremented.

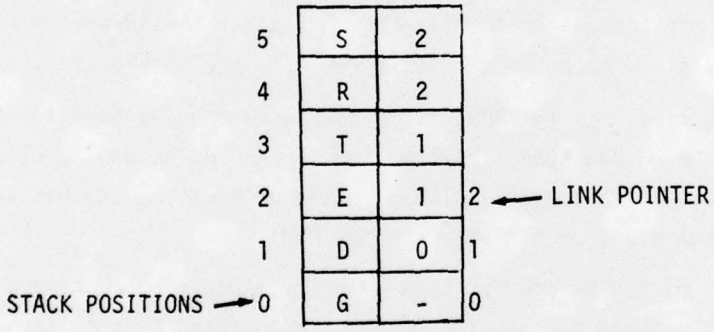


Figure 6-31. Path Stack

During path search, the addresses for the PS come from STP which is incremented each time the stack gains a new member. However, once the destination node has been placed on the stack, the path threading operation commences and the addresses then come from the link fields. Thus, if "S" had been the destination node, the stack address of the next member of the path would come from S's link field, viz., 2. E's link field points to D, who in turn points to G. Thus, the path would be given as G-D-E-S.

The threaded path stack (TPS) is loaded with the local names of the nodes making up the completed path. In the example above, the nodes placed on the TPS would be S, E, D, and G in that order. At this point, the sequential controller for the path selection hardware would notify the node controller that a path has been formulated.

Path Selection Controller. The control functions required by the path selection hardware are implemented using a restricted type of Moore machine in which there are only two possible exits from any state. The detailed schematic of the hardware is shown on Figure 6-26. The controller is in the top left-hand corner of the drawing.

The controller has three basic parts: read-only memory, state counter, and a multiplexer. There are three types of output signals from the ROM: control lines which are routed to the various path selection chips, address lines which define the next state address in the event of a jump, and multiplexer control lines which are used to select which input variable is monitored to determine if the jump is to be taken.

The number of bits in the state counter are found by taking the ceiling of the $\log_2 n$, where n is the number of states in the state diagram (see Figure 6-32). The state diagram for the path selection algorithm uses 22 states, thus the state counter has $\lceil \log_2 (22) \rceil$ bits,* i.e., 5 bits. This counter is preset to the value of the next address lines when implementing a jump in the state diagram, or it can be cleared to all zeroes, which is used as an entry point to the state diagram. State 8 of Figure 6-32 depicts a jump condition. If PF is true, then the state counter is incremented and state nine will be executed next. However, if PF is false ($\overline{PF}$ true), then the next address lines (encoded with the binary expression for 17) are strobed into the state counter and state 17 will be the next executed state. The use of state zero as an entry point is shown in the bottom left-hand corner of the state diagram. Note that state zero can be entered at any time by pulsing the CLEAR input on the state counter. In this case, clearing the state counter would initiate the PMR refresh cycle after which the controller would enter state four and wait for a START command.

The number of lines entering the multiplexer depends on the number of conditional signals present in the hardware. As can be seen from the state diagram, there are five conditionals: CRYSPM, START, PF, A=B, and A<B. In actuality, there are two more conditionals, "1" and "0". The "1" conditional is used for the situation in which a jump always occurs. For example, the transition from state 21 to state 6 or the transition from state 19 to state 4. These jumps occur because of loops in the flow of control. The "0" conditional is used for nonjump situations, e.g., the transition from state 6 to state 7. This input to the multiplexer could be eliminated

*$\lceil X \rceil$ is the ceiling of X, i.e., the smallest integer greater than or equal to X

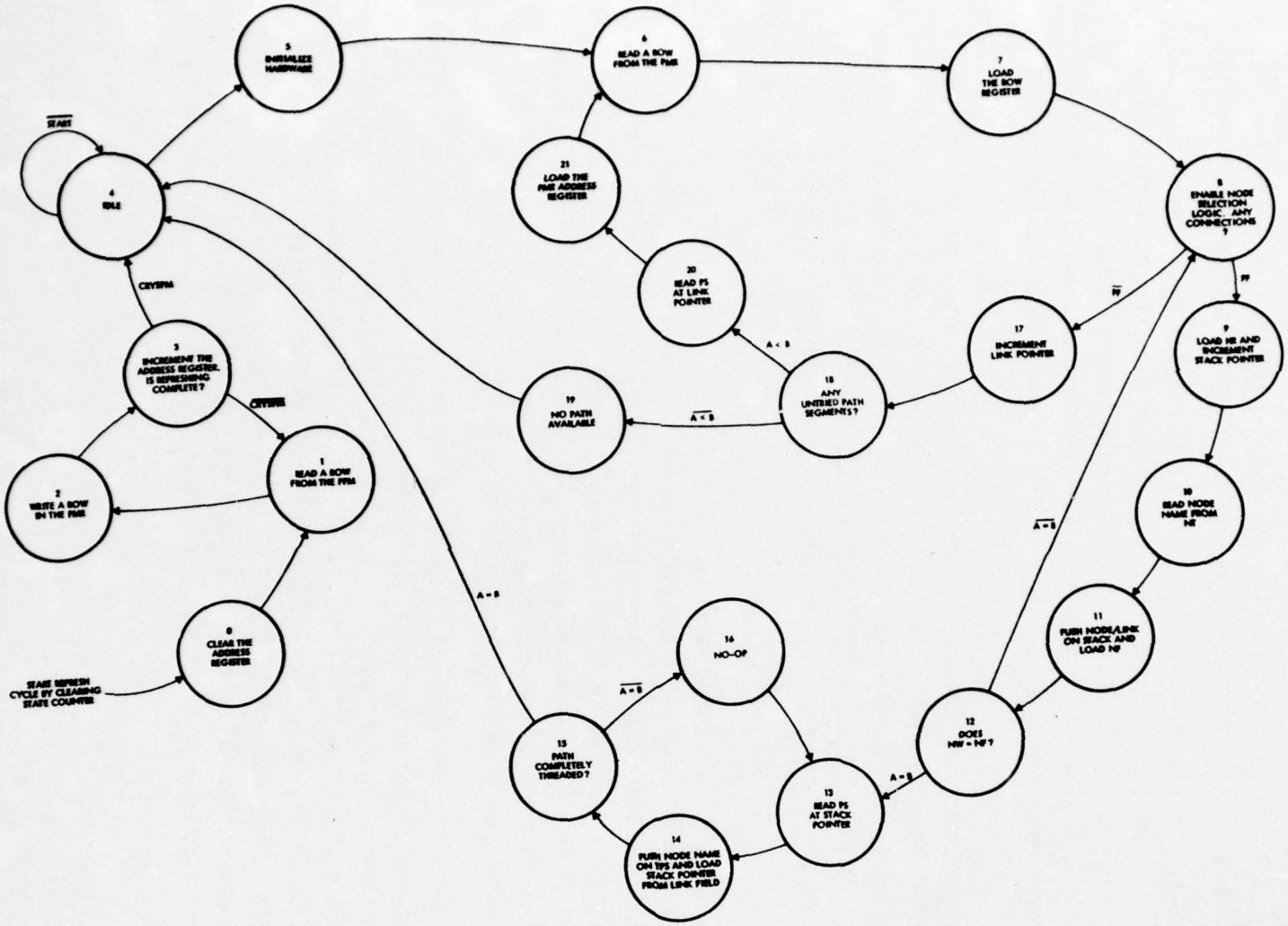


Figure 5-32. Sequential Controller State Diagram

by using an extra ROM output to enable the multiplexer only for jump situations. However, in this case an eight-input multiplexer will be used so that the extra input is already available. The number of mux control lines needed can then be calculated with the same formula used for finding the number of bits in the state counter, where n is now the number of input lines to the mux instead of being the number of states in the state diagram.

The number of bits needed in the ROM can be reduced somewhat by storing the control signal information in an encoded form and using n-to-2^n decoders. For example, if we have four control signals and each signal occurs during different states, they can then be encoded with two bits. These two bits are then decoded using a 2-to-4 decoder to yield the four control signals. Using this technique, the number of bits needed for the control signals were reduced from 37 to 14.

The detailed flowchart for the controller for the path selection algorithm is presented in Figure 6-33. All the element names (e.g., PS, PMR, etc.) are taken from the logic diagram (Figure 6-26). There are three main sections in the flowchart: the first section deals with refreshing the scratchpad version of the path matrix, the second section describes path search, while the third deals with path threading.

The scratchpad path matrix (PMR) is renewed by simply copying into it the contents of the permanent path matrix (PPM). This is done in four steps: read a word from the PPM, write that word into the PMR, increment the address register (SPM), and check to see if the process is complete. The last two operations are performed essentially at the same time. When the address register is incremented, carry out (CRYSPM) is checked for a "1" which would signify the end of the refresh cycle. If CRYSPM is a "0", then the next word is read from the PPM, etc. Having generated a fresh copy of the path matrix, the controller sits in a wait loop (state 4) until a request for a path search is received.

When a path search is finally requested (START=1), the controller initializes the hardware by clearing address registers SPM and STP, the link field pointer (LKP), and the threaded path stack (TPS). It also strobes the destination global node name into the NW register and sets the used word register (UW) to all ones.

It is assumed that row zero in the path matrix represents connections to the node at which the path is being formulated. Thus, clearing SPM ensures that the first nodes nominated as potential path members will be one branch away. Each bit in the used word register enables the corresponding bit in the RR. If the UW bit is high, then the RR bit will be loaded from the chosen row in the scratchpad path matrix. However, if the UW bit is zero, indicating that the node associated with that particular bit has already been pushed on the stack, then the RR bit would be held at zero.

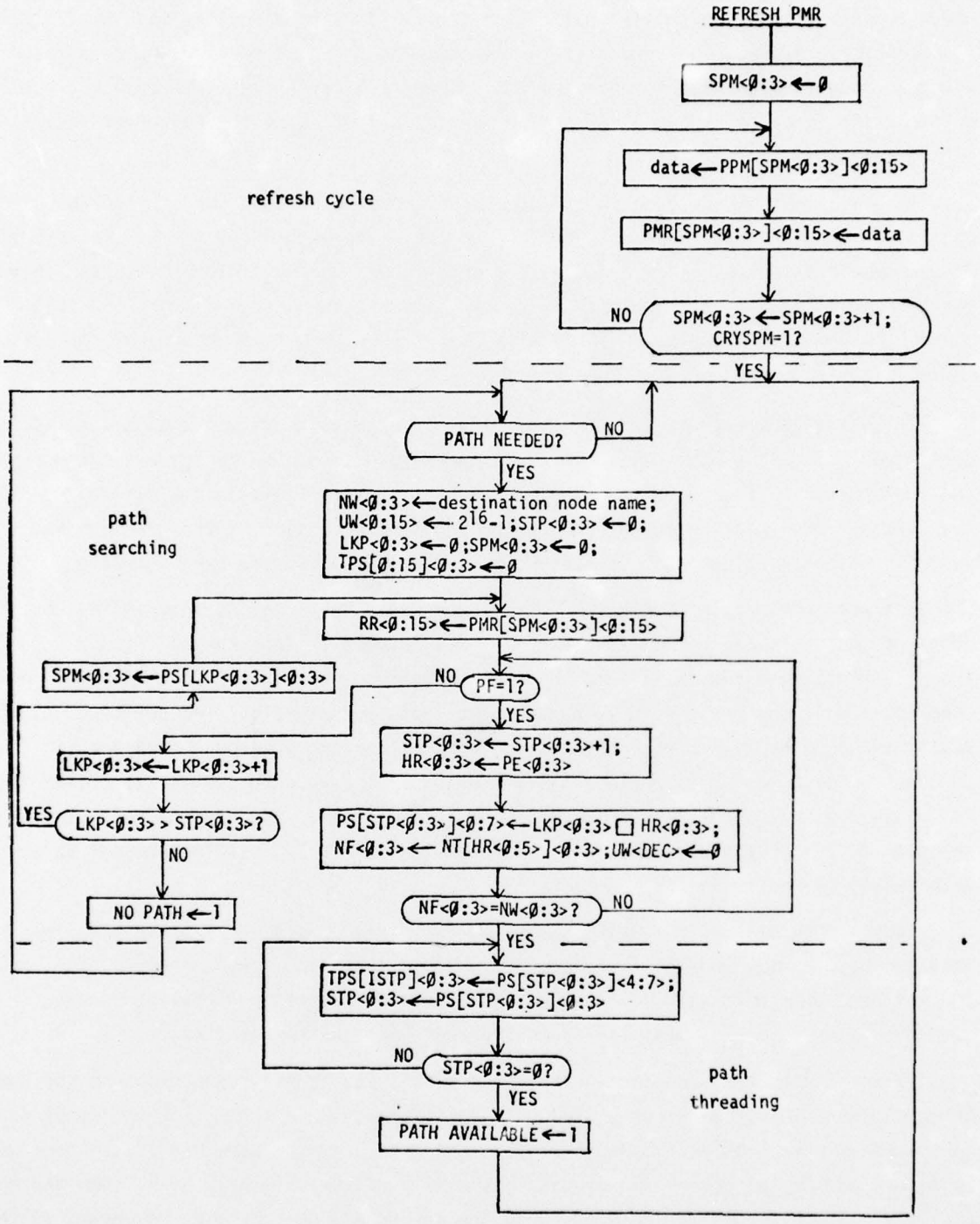


Figure 6-33. Sequential Controller Flowchart

At this point, the node selection logic is enabled. This set of gates locates the left-most "1" in the word held by the row register and decodes the position of that bit in the word (which is the local name for that particular node). If the row is all zeroes, the priority flag (PF) will remain at zero. Assuming that a node has been found, the 4-bit local name is strobed into the holding register (HR). The contents of the HR are sent to a 4-line-to-16-line decoder whose outputs are used to clear the bit associated with that node in the UW register. They are also used to fill the node name field of the stack word and as an address for the name table to translate the local node name into the global node name (note that the global node name does not have to be limited to 4 bits; however, if it is greater than 4 bits, the NW, NF, and comparator circuit have to be extended to accommodate the larger word size).

Upon finding a connection, the stack pointer (STP) is incremented and the new stack word, which is a concatenation of the contents of the HR and link pointer registers, is loaded into the path stack. Once this has been accomplished, the NW and NF registers are compared to find out if the last node included in the stack was the destination node. If it was not, then the node selection logic is enabled again in hopes of finding another candidate. If the destination node has been reached, then the path must be threaded and stored in the TPS.

When the destination node has been located, the STP register still points to the stack at the destination node position. The PS is then read at this address with the results that the node name field is stored in the threaded path stack while the link field is strobed into the STP register. This process continues until the link field is null (zero), which indicates that the complete path has been stored in the TPS. The node controller is notified of the existence of a path when PATH AVAILABLE is raised by the path selection controller.

If the row register contains all zeroes, which occurs when all the candidates connected to the point-of-search node have been included in the PS, then the link pointer is incremented and compared to the stack pointer. If LKP is greater than STP, then a path is impossible because all possible candidates have been nominated. In this case, the NO PATH flag is raised informing the node controller of this unfortunate situation. If LKP is less than or equal to STP, then the name of the next point-of-search node is read from the stack and its row in the path matrix is checked for new, unused candidates.

Recommendations for Future Work

The path selection logic appears amenable to integration with the exception of the permanent path matrix and name table ROM. The controller implementation might be greatly simplified with use of the recently announced Fairchild I^3L microprogram

sequencer.* In addition to assessing the impact of this new product, a complete register transfer level simulation should be performed to verify correct operation prior to commencing the integration effort. In addition to verifying correct operation of the logic a register transfer simulation greatly simplifies the development of test methods.

*R.E. Crippen, et al, "Microprogram Sequencer, Utilizing I^3L Technology," Proc. IEEE ISSCC, Vol. 19, 1976, pp. 98-99.

7. RECOMMENDATIONS FOR FUTURE WORK

With the study phase and the first phase of a projected three phase technology development effort an accomplished fact, the program goals and objectives originally set forth by the Naval Research Laboratory and TRW remain unchanged. Based on the encouraging experimental results to date, the next phase should be the experimental verification of the computational functions required for the demonstration vehicle to follow. No deviation from the original plan is required.

Initiation of an effort to investigate and characterize in a radiation environment the digital CCD circuitry being developed is recommended. This effort should be directed by the Naval Research Laboratory; their experience in this area will be extremely beneficial. The beginning effort should be one of determining the particular characteristics of digital CCD's in the anticipated environment. The differences in design and operation are likely to generate different responses to radiation in the digital devices compared to analog CCD's. Having made these measurements, the results can then be used to gauge what appropriate action should be taken to improve the radiation resistance of the device. This work should begin as soon as possible. The first computational device array is available from the DP-1 test devices. Device performance measurements should be made before future arrays are further developed.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. 3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (and Subtitle) (6) Charge Coupled Devices in Signal Processing Systems. Vol. III. Digital Function Feasibility Demonstration. (Volume)	5. TYPE OF REPORT & PERIOD COVERED Rept. for Jan 75 - Feb 76 pix
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Thomas A. Zimmerman, Roland J. Handy, Reginald A. Allen, Earl E. Swartzlander, Jr. Douglas J. Heath, J. E. Sandor	8. CONTRACT OR GRANT NUMBER(s) (15) N00014-74-C-0068
9. PERFORMING ORGANIZATION NAME AND ADDRESS TRW Systems Group Redondo Beach, California	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Research Laboratory (11)	12. REPORT DATE March 1976
	13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 145 p.	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Charge Coupled Device	Digital Multipler	Distributed Networks
CCD	Digital Adder	
Digital Functions	Fast Fourier Transform	
Full Adder	Transparent Interface	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In 1973 the Navy began a program to investigate and demonstrate the impact of the use of charge coupled devices (CCDs) in the digital domain.

This report summarizes the digital function feasibility demonstration phase of that program.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

354 595

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LB

The concept of digital signal processing with CCDs is discussed briefly. Two implementations of an adder circuit are described and the development of each implementation in arrays to perform arithmetic operations is shown. The experimental results derived from these individual adders and the overall arrays is presented along with some theoretical models.

Also presented are the results of a study to determine an efficient means of interconnecting a number of independent processors; the digital CCD development program is expected to produce such independent data processors contained on one or two chips. The study describes a method of interfacing an arbitrary number of such processors that employs distributed network control and exhibits soft failure.

Recommendations are included for future work on all the aspects treated in this volume.